REPORT DOCUMENTATION PAGE

	1b. RESTRICTIVE MARKINGS
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) FR-11Jul90	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 90 0866
6a. NAME OF PERFORMING ORGANIZATION Prometheus Inc. / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c. ADDRESS (City, State, and ZIP Code) 103 Mansfield St. Sharon, MA 02067	7b. ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB DC 20332-6448
8a. NAME OF FUNDING / SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-88-C-0028
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
New Approaches to Beamforming, Null Steering, and Filtering (Unclassified)

12. PERSONAL AUTHOR(S) Barrett, Benedetto, Boyd, Byrnes, Giroux, Khraishi, Miller, Newman, Ostheimer, Roy, Shapiro

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 881201 TO 900331	90, July, 11	207

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	beamforming, null steering, filtering, shading coefficients, antennas, convex programming, arrays, peak factor, wavelets, spectrum estimation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This final report consists of ten sections. The first is a report describing our abilities in the co-site radio frequency interference (RFI) problem. The second is the paper *Heisenberg Wavelets and the Uncertainty Principle*, which will be expanded upon and submitted to the appropriate technical journal. The third is a revised version of the paper *A Multidimensional Wiener-Wintner Theorem and Spectrum Estimation*, which will appear shortly in the Transactions of the American Mathematical Society. The fourth is the expository paper *Problems on Polynomials with Restricted Coefficients Arising from Questions in Antenna Theory*, to appear shortly in the Proceedings of the 1989 NATO Advanced Study Institute on Fourier Analysis and its Applications. The fifth is the paper *An Ideal Omnidirectional Transmitting Array, and Optimal Peak Factor Array, for Less Than Half-wavelength Spacing*, to be submitted shortly to the IEEE Transactions on Antennas and Propagation. The sixth is the revised paper *Properties on the Unit Circle of Polynomials with Unimodular Coefficients*, which just appeared in the Proceedings of the American Mathematical Society. The seventh gives an overview of awd, which is our new method of choosing shading coefficients, based on a convex programming approach. The eighth is the latest revision of the paper *A Specification Language Approach to Solving Convex Antenna Weight Design Problems*, which will be submitted shortly to the appropriate technical journal. The ninth is version 3.0 of the awd *User's Manual*, incorporating several improvements which have been added recently. Several specific awd examples which were done for the Naval Underwater Systems Center constitute the tenth and final section.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

New Approaches to Beamforming, Null Steering, and Filtering

Final Report 11 July 1990

Prometheus Inc.
103 Mansfield Street
Sharon, MA 02067

Contract # F 49620-88-C-0028

Prepared by:

Michael J. Barrett, John J. Benedetto, Stephen Boyd,
James S. Byrnes, Andre Giroux, Nasser M. Khraishi, Kenneth S. Miller
Donald J. Newman, Gerald Ostheimer, Richard Roy, Harold S. Shapiro

Submitted to:

Dr. Arje Nachman
Mathematics and Information Sciences
Air Force Office of Scientific Research

Submitted by:

James S. Byrnes

James S. Byrnes, President

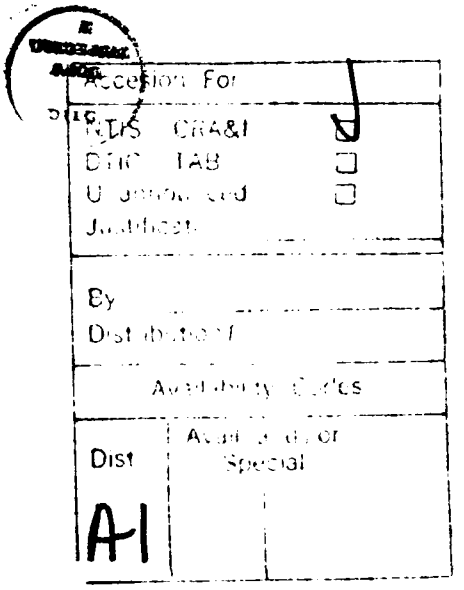
11 July 1990

Date

New Approaches to Beamforming, Null Steering and Filtering

Abstract

This final report consists of ten sections. The first is a report describing our abilities in the co-site radio frequency interference (RFI) problem. The second is the paper *Heisenberg Wavelets and the Uncertainty Principle*, which will be expanded upon and submitted to the appropriate technical journal. The third is a revised version of the paper *A Multidimensional Wiener-Wintner Theorem and Spectrum Estimation*, which will appear shortly in the Transactions of the American Mathematical Society. The fourth is the expository paper *Problems on Polynomials with Restricted Coefficients Arising from Questions in Antenna Theory*, to appear shortly in the Proceedings of the 1989 NATO Advanced Study Institute on Fourier Analysis and its Applications. The fifth is the paper *An Ideal Omnidirectional Transmitting Array, and Optimal Peak Factor Array, for Less Than Half-wavelength Spacing*, to be submitted shortly to the IEEE Transactions on Antennas and Propagation. The sixth is the revised paper *Properties on the Unit Circle of Polynomials with Unimodular Coefficients*, which just appeared in the Proceedings of the American Mathematical Society. The seventh gives an overview of **awd**, which is our new method of choosing shading coefficients, based on a convex programming approach. The eighth is the latest revision of the paper *A Specification Language Approach to Solving Convex Antenna Weight Design Problems*, which will be submitted shortly to the appropriate technical journal. The ninth is version 3.0 of the **awd** *User's Manual*, incorporating several improvements which have been added recently. Several specific **awd** examples which were done for the Naval Underwater Systems Center constitute the tenth and final section. The report concludes with the Appendix, which contains various miscellaneous information.



Prometheus Inc.
Final Report
11 July 1990

Section I
Co-site RFI Problem

On Calculating the Effects of Nonlinearities in Broadband RF Communication Systems

M. J. Barrett, K. S. Miller and Richard Roy
Prometheus Inc.
21 Arnold Avenue
Newport, RI 02840
(401) 849-5389

September 2, 1989

1 Introduction

This document is to inform the clients of Prometheus Inc. of our interest and abilities in the co-site radio frequency interference (RFI) problem. This problem has arisen due largely to the tremendous difficulty and concomitant lack of experience and practice in the simulation of mutual interference between transmitters and receivers caused by system nonlinearities. A special simulation technology has been developed to handle this problem.

Given a particular transceiver (co-sited transmitter and receiver) and set of transmitter waveforms as well as a mathematical description of nonlinearities present, the simulation technology developed predicts the accumulated RFI in specified receiver bandwidths. This is obviously useful in the design of transceivers to minimize RFI due to system nonlinearities. In situations where the design of the system is fixed and RFI reduction is of importance, the simulation technology provides estimates of the dominant sources of RFI. This is valuable information in attempting to arrive at cost effective solutions to the problem. The actual solutions are, of course, problem dependent and

can incorporate temporal adaptive filtering techniques as well as sensor array processing techniques that take advantage of spatial diversity of the various RFI sources.

2 Background

There are many practical applications in which the ability to simultaneously transmit and receive RF signals using a single antenna system is desirable. Ideally, though physically co-located, the two functions, transmission and reception, can be separated in the frequency domain so as to prevent mutual interference. However, nonlinearities in the output stages of the transmitters and in the input stages of the receivers, and nonlinear mixing effects in the platform environment can lead to *co-channel* interference. These nonlinear problems are difficult for most engineers to understand and predict, and are particularly difficult for those engaged in writing EMI specifications.

Co-site interference is associated primarily with several nonlinear effects. Undesirable nonlinear mixing of signals from different sources can occur in the *skin* of the platform and in the front-ends of receivers. Broadband self-noise associated with switching transients in the frequency synthesizers of spread-spectrum systems is mixed with itself in the output stages of transmitters, since microwave amplifier output stages can often be quite nonlinear. These effects cause the received signal-to-interference ratio to deteriorate as the transmitted signal passes through the output stage. In addition, intermodulation products and noisy harmonics generated in nonlinear mixing from various external sources can appear in the *wrong* bands, causing further deterioration of the performance of co-sited systems.

It is generally accepted that the current difficulty in understanding and coping with co-site interference between transmitters and receivers is partly due to a lack of simulation models for the calculation/prediction of intermodulation and harmonic interference. In addition to the inherent mathematical complexity of nonlinear systems, the generation of efficient computer models for these systems is made even more difficult by the very great bandwidths and frequency separations involved in EMI between such systems.

3 Predicting Nonlinear Effects

Volterra-Wiener analysis [1] is probably the best-known technique for handling nonlinear systems, although it has not lived up to its promise as a means of predicting nonlinear effects in communication systems. One of the major difficulties is that Volterra transfer functions [VTFs] (or Volterra *kernels*) for nonlinear amplifiers with memory grow in dimension with the order of each term in a power series. That is, the second-order VTF is a two-dimensional function (a complex-valued function on a two-dimensional real frequency domain), the third-order VTF is three-dimensional, *etc.*. Due to the exponentially increasing complexity, Volterra series are generally truncated at orders of around four or five.

In general, the computational complexity involved in approximating broadband nonlinear systems with *fading memory* by Volterra series is prohibitive. For *memoryless* or *instantaneous* nonlinearities, however, the computational complexity is greatly reduced. The resulting approximation can be viewed as a Taylor series expansion of the nonlinearity at the origin. As a consequence, the issue of convergence of the series must be addressed. For example, non-analytic memoryless nonlinearities do *not* have Volterra series representations. The nonlinearities must first be approximated by analytic functions, adding another degree of complexity and approximation.

The new technique proposed herein involves expansion of the instantaneous nonlinearities in a set of polynomials orthogonal with respect to the complex Gaussian measure. It differs from Volterra expansions most significantly in that analytic functions are not required; tabular I/O characteristics of the instantaneous nonlinearities can be handled. When applied to a single amplifier, the new technique also differs considerably from the analysis of Fuenzalida *et. al.* [2], wherein nonlinear amplifier characteristics are expanded in a series of Bessel functions. But it has in common with that analysis the ability to accurately model both the amplitude modulation/amplitude modulation [AM/AM] and amplitude modulation/phase modulation [AM/PM] characteristic curves. In fact, these may be input to the simulation in equal decibel increments rather than in volts.

The alternative approach developed by Prometheus Inc. is specifically tailored to the RFI and nonlinear amplifier problem. It incorporates engineering models designed for the purpose of generating intermodulation effects in a computer representation which is suitable for, and compatible with, existing

computer simulations of military communication and radar systems. The simulations can accurately predict the interference effect of one system, or group of systems, upon another.

The most difficult problem in simulating communication systems is the sample rate required. For instance, a typical commercial satellite transponder has a bandwidth of 40 MHz, which requires complex sampling at 20 MHz to prevent aliasing after down-conversion to baseband. Now consider simulating a combination of two such transponders separated by 100 MHz. It would be unthinkable to simulate these systems in the actual combined bandwidths they occupy, as this would require sample rates in excess of 100 MHz. Where multiple signal spread-spectrum systems are involved, the overall spread-spectrum bandwidth of a given system is generally too great to consider simulating directly. Such systems are usually simulated in the de-spread baseband bandwidth of one signal, calculating where necessary the effects of the spreading and de-spreading processes on the other signals involved.

The new simulation technique allows the generation of *all* of the intermodulation products that fall into a given band from a group of systems in several widely spaced frequency bands. The simulations can be carried out in sample rates appropriate to the transmission bandwidths of each system, referred to baseband *in-phase* and *quadrature* [IP&Q] samples.

4 Example

By way of introduction to the new technique, let us suppose that a metal fastener, bolt, or rivet on the surface of the communications platform has become an undesirable RF radiator due to a metallic interface that acts like a half-wave rectifier. The rectifier causes mixing of signals from two transmitters, resulting in intermodulation products that fall into the band of a co-located receiver. An adequate model for the purpose at hand would be to sum the two signals and pass the sum through an instantaneous nonlinearity, the actual characteristics of which could be determined experimentally.

For the purposes of computer simulation, adding the two signals together at RF and performing the indicated operation is not practical because the bandwidth and associated sample rates are astronomical. It is preferable to carry out simulations in the baseband bandwidth of each interfering system, and to generate intermodulation products as they appear in the baseband of

the system that is being interfered with. Thus, data are required only in the conventional IP&Q baseband format (compatible with existing simulations of the systems).

In the hypothetical problem, there are two inputs and one output. The output band of interest is only a small fraction of the entire bandwidth of intermodulation products resulting from the nonlinear mixing of the two signals. Furthermore, the actual carrier frequencies are not present in the data (since all data is referenced to baseband). Therefore, a means of keeping track of the sum and difference frequencies between carriers that set up the entire pattern of intermodulations is required.

The nonlinear computer-aided analysis and simulation technique that was recently developed [3] for the analysis and simulation of nonlinear effects in microwave amplifiers satisfies these requirements. Any complex-valued (instantaneous) nonlinear function of a complex input or inputs can be represented, and the carrier frequencies *factored out*, so that mixing between two close, widely spaced, or overlapping radio frequency bands can be done with complex baseband IP&Q samples. There are no *rule-of-thumb* approximations involved, and computations may be carried out to any desired degree of accuracy. The technique is limited only by the degree of accuracy of the model of the nonlinearity. It can be readily demonstrated that all microwave nonlinearities of interest here are sufficiently *instantaneous* to satisfy the fundamental requirement. The technique will produce output for only the desired band of the system that is being interfered with, without the need for generating intermodulation products that are outside the bandwidth of interest.

The new technique is based upon a nonlinear input-output theory developed by Price [4]. Like the Volterra-Wiener analysis, it was originally intended for random inputs, but is suitable for deterministic inputs. It uses expansions of the nonlinear instantaneous transfer characteristics in complex Hermite polynomials, rather than in a power series in the input like the VTFs and Wiener G-functionals. A brief discussion of complex Hermite polynomials is given in the Appendix. Again, the interest here is not in Gaussian random process inputs, but rather in actual non-Rayleigh envelope signals. Unlike previous traveling wave tube [TWTA] analyses, there is no attempt here to determine output statistical properties. The main objective is to provide a detailed simulation of the actual signals.

5 Conclusion

The Prometheus team has already successfully applied complex Hermite polynomial expansions to microwave amplifier saturation characteristics, and demonstrated excellent curve fitting over the 20 dB range of the Rayleigh decibel density. In that application, actual saturation characteristic curves were fit with complex Hermite polynomials of orders less than ten. Close phase shift and gain agreement between the polynomial approximations and the corresponding actual AM/AM and AM/PM saturation curves were achieved. The same technique can be used with more general nonlinearities, such as microwave rectifiers, and will accurately predict the outputs with nothing other than simple numerical truncation. As much accuracy as desired can be implemented.

6 Discussion

The capability described in this document provides a computationally feasible means for predicting the effects of various instantaneous nonlinearities on the performance of co-sited RF communication systems. It is a valuable tool in the design and analysis of such systems. In the design stage, the susceptibility of the performance of a particular communication system design to various types of nonlinearities can be predicted. An area for future work would be the inclusion of this capability into an automated design facility, where optimum system parameters minimizing performance degradation are sought.

In the system analysis and debugging phase of communication system design, the tool is capable of predicting, given several nonlinearities, those that are contributing most prominently to the performance degradation. Consequently, those nonlinearities that should be eliminated first in an attempt to improve performance can be identified. Similarly, the tool can be used to identify nonlinearities that are present in a particular physical system in the experimental phase. Inputs can be designed to *isolate* certain types of nonlinearities, and actually applied to the system. The presence of certain components in the output can then be attributed to the particular nonlinearity in question.

Having isolated the potential sources of the performance degradation,

elimination of the problem can then be addressed. In the situation where an array of receivers is present and the significant RF nonlinearities are point source radiators, recently developed techniques in the area of sensor array signal processing can be employed to significantly reduce the effect of the nonlinearities. The basic idea involves spatially isolating the disturbance source, and employing a transformation of the sensor array outputs (receiver inputs) that eliminates the disturbance.

A Mathematical Development

In the simple example described previously, a half-wave rectifier causes mixing of signals from two transmitters, resulting in intermodulation products that fall into the band of a co-located receiver. The appropriate mathematical model involves passing the sum of the two signals through an instantaneous nonlinearity, the actual characteristics of which could be determined experimentally.

$$\tilde{z}_1 + \tilde{z}_2 = z_1 e^{j\omega_1 t} + z_2 e^{j\omega_2 t} \tag{1}$$

The tilde represents radio frequency (RF) signals and z_1 and z_2 are the complex baseband signals. Only the *pre-envelope* or positive-frequency representations of the RF signals are required. The RF signal is now rectified by an instantaneous nonlinearity:

$$y = f(\tilde{z}_1 + \tilde{z}_2) \tag{2}$$

The instantaneous nonlinearity is assumed to be a complex-valued function, $f(z)$, of a complex variable. For example, an ideal half-wave rectifier is described by

$$f(\tilde{z}) = \begin{cases} \tilde{z}, & \text{Re } \tilde{z} > 0 \\ 0, & \text{Re } \tilde{z} < 0 \end{cases} \tag{3}$$

in these representations. Note that the function need not be analytic.

The basic idea is to expand the nonlinear function in a series of complex Hermite polynomials $H_{n,m}$:

$$f(\tilde{z}) = \sum_{n,m} K_{n,m} H_{n,m}(\tilde{z}, \tilde{z}^*). \tag{4}$$

Complex Hermite polynomials appear very infrequently in the literature, and thus are not well known, even to mathematicians. They are closely associated with the complex normal density in the same way that ordinary Hermite polynomials are associated with the normal density. In each case, a probability density is the weight function of a set of orthogonal polynomials. The use of the complex Hermite polynomials versus another suitable set of orthogonal complex polynomials is somewhat arbitrary in this application, although they can be shown to have suitable curve-fitting properties for the problem at hand, as they do for TWTA characteristics.

Noting that the complex Hermite polynomials are orthogonal polynomials with a weight function which is a (unit-variance) complex normal probability density, the coefficients in the expansion may be determined by:

$$K_{n,m} = c(n,m) \int \exp(-\frac{1}{2}|\tilde{z}|^2) f(\tilde{z}) H_{n,m}(\tilde{z}^*, \tilde{z}) d\tilde{z}, \tag{5}$$

where $c(n,m)$ is a normalizing factor, and it is understood that the differential elements are the real and imaginary parts of z rather than the complex variable itself. The integral is actually a two dimensional integral in real variables.

The complex Hermite polynomials form a *pyramid* of the type as indicated in Figure 1 with those along the outside border of the pyramid of the

$$\begin{array}{ccccc} & & 1 & & \\ & & \tilde{z} \;\; \tilde{z}^* & & \\ & \tilde{z}^2 & \cdot & \tilde{z}^{*2} & \\ \tilde{z}^3 & & \cdot\;\cdot & & \tilde{z}^{*3} \\ & & \cdots & & \end{array}$$

Figure 1: Pyramid of Complex Hermite Polynomials

form $\tilde{z}^n$ or $\tilde{z}^{*n}$. The *internal* polynomials lead with the term $\tilde{z}^n \tilde{z}^{*m}$ which is indicated by that particular polynomial's position on the pyramid. Every term of a particular polynomial will retain the difference between the number of powers of z and z^* that the leading term has. The expansion of an amplitude sensitive nonlinearity, like the TWTA, uses only those polynomials along the *diagonal* for which there is one more power of z than of z^*. This occurs because a purely amplitude sensitive nonlinearity does not generate harmonics [5], but only intermodulations in or near the original band

of transmission. The *half-wave rectifier*, on the other hand, has a general expansion using terms from all of the *diagonals*.

The carrier frequencies can now be *factored-out*, and the output can be obtained in only a desired band. In any particular Hermite polynomial, the difference between the number of powers of z and of its complex conjugate z^* is fixed. Using (1), (2), and the binomial theorem:

$$H_{n,m}(\tilde{z},\tilde{z}^*) = \sum_x \text{coeff.} \times (z_1 e^{j\omega_1 t} + z_2 e^{j\omega_2 t})^n (z_1^* e^{-j\omega_1 t} + z_2^* e^{-j\omega_2 t})^m \quad (6)$$

$$= \sum_x \text{coeff.} \times \sum_{p=0}^{n} \binom{n}{p} (z_1 e^{j\omega_1 t})^p (z_2 e^{j\omega_2 t})^{n-p} \quad (7)$$

$$\times \sum_{q=0}^{m} \binom{m}{q} (z_1^* e^{-j\omega_1 t})^q (z_2^* e^{-j\omega_2 t})^{m-q} \quad (8)$$

The carrier frequencies ω_1 and ω_2 can be factored-out with their various multiplicities, leaving only the baseband signals z_1 and z_2, with their intermodulations, for the numerical simulation to contend with. But, by keeping track of the carrier frequencies as a side problem, it is readily evident where the intermodulations belong in the frequency spectrum. A particular term would be, for instance, of the form:

$$z_1^p z_2^{n-p} z_1^{*q} z_2^{*m-q} \exp[jt[(p\omega_1 + (n-p)\omega_2 - q\omega_1 - (m-q)\omega_2]] \quad (9)$$

The sums and differences of the carrier frequencies may be accounted for, and the frequency of the intermodulation band specified. Only the desired band of intermodulation products need be simulated. The corresponding coefficients in the complex Hermite polynomial expansion may be determined in advance of the simulation.

References

[1] S. Narayanan, Transistor distortion analysis using Volterra series representation, *The Bell System Technical Journal*, pages 991–1024, May–June 1967.

[2] J. C. Fuenzalida, O. Shimbo, and W. L. Cook, Time-domain analysis of intermodulation effects caused by nonlinear amplifiers, *Comsat Technical Review*, **3**(1):89–143, 1973.

[3] M. J. Barrett, Nonlinear analysis of travelling wave tube amplifiers using complex Hermite polynomials, Preprint.

[4] W. F. McGee, Circularly complex Gaussian noise – A Price theorem and a Mehler expansion, *IEEE Trans. Inform. Theory*, **IT-5**:317–319, March 1969.

[5] J. H. van Vleck and D. Middleton, The spectrum of clipped noise, *Proc. IEEE*, **54**(1):2–19, 1966.

Prometheus Inc.
Final Report
11 July 1990

Section II
Heisenberg Wavelets

Heisenberg wavelets and the uncertainty principle

John J. Benedetto*
Prometheus Inc.

For the past few years the term "wavelet" has indicated a function $\psi \in L^2(\Re)$ for which the set $\{\psi_{m,n}\}$ of dilations and translations, associated with the affine group, is an orthonormal basis of $L^2(\Re)$. In light of the theory of wavepackets and the sustained interest in Weyl-Heisenberg decompositions because of the importance of frequency localizaton, we refer to both "affine wavelets" ψ and "Heisenberg wavelets" $\phi \in L^2(\Re)$, where

$$\phi_{m,n}(t) = e^{2\pi imbt}\phi(t - na) \tag{1}$$

for fixed $a, b > 0$. The term "Heisenberg" is used because of the underlying importance of the Heisenberg group in analyzing (1). Elementary properties of $\{\psi_{m,n}\}$ and $\{\phi_{m,n}\}$ are given in [Ben89, HW89]. This note concerns Heisenberg wavelets.

The construction of Heisenberg wavelets is a traditional goal in signal processing, e.g., Gabor's classical paper [Gab46]. The idea originated in quantum mechanics where "over complete" sets (coherent states) are associated to objects such as the Weyl-Heisenberg group or the von Neumann lattice. Related devices include the Wigner distribution (1932) and the ambiguity function. The Wigner distribution became a tool in signal analysis in the 1940's (by Ville) and presently plays an important role in a host of electrical engineering problems [CM80]. The ambiguity function is a staple in optics and high resolution radar.

1 The uncertainty principle inequality.

The uncertainty principle inequality is

$$\|f\|_2^2 \leq 4\pi\|(t - t_0)f(t)\|_2\|(\gamma - \gamma_0)\hat{f}(\gamma)\|_2. \tag{1.1}$$

The Fourier transform $\hat{f}$ is defined as

$$\hat{f}(\gamma) = \int f(t)e^{-2\pi it\gamma}\, dt,$$

where $t \in \Re, \gamma \in \hat{\Re}$ (=$\Re$), and integration is over $\Re$. Our notation is standard, e.g., [SW71], and (1.1) is the simplest of a whole class of uncertainty principle inequalities, e.g., [Ben90].

There is equality in (1.1) in the case f is the Gaussian. Because of this, one of Gabor's ideas from [Gab46] was to write discrete expansions of $L^2(\Re)$ elements in terms of $\{\phi_{m,n}\}$, where ϕ is the Gaussian. This part of his program is subject to some scrutiny because of the Balian-Low theorem: If $\{\phi_{m,n}\}$ is a frame for $\phi \in L^2(\Re)$ and $ab = 1$ then either $t\phi(t) \notin L^2(\Re)$ or $\gamma\hat{\phi}(\gamma) \notin L^2(\Re)$ [BHW].

---

* The author is also Professor of Mathematics at the University of Maryland, College Park, MD 20742.

2 Weak uncertainty.

Given $\psi \in L^2(\Re)$ for which $\|\psi\|_2 = 1$. It is well known and easy to verify that

$$\|(t - t_{m,n})\psi_{m,n}(t)\|_2 \|(\gamma - \gamma_{m,n})\hat{\psi}_{m,n}(\gamma)\|_2 = \|(t - t_{0,0})\psi(t)\|_2 \|(\gamma - \gamma_{0,0})\hat{\psi}(\gamma)\|_2, \qquad (2.1)$$

where the *expected values* $(t_{m,n}, \gamma_{m,n})$ are

$$t_{m,n} = \int t|\psi_{m,n}(t)|^2\, dt \quad \text{and} \quad \gamma_{m,n} = \int \gamma|\hat{\psi}_{m,n}(\gamma)|^2\, d\gamma,$$

e.g., [Ben90].

An analogous result is true for Heisenberg wavelets—

Proposition 2.1. *Given $\phi \in L^2(\Re)$ for which $\|\phi\|_2 = 1$ and let $ab = 1$. Assume that $t\phi(t), \gamma\hat{\phi}(\gamma) \in L^2$ and*

$$\int t|\phi(t)|^2\, dt,\ \int \gamma|\hat{\phi}(\gamma)|^2\, d\gamma \in \Re.$$

Define the variances

$$\Delta^{t2}_{m,n} = \|(t - t_{m,n})\phi_{m,n}(t)\|_2^2 \quad \text{and}$$
$$\Delta^{\gamma 2}_{m,n} = \|(\gamma - \gamma_{m,n})\hat{\phi}_{m,n}(\gamma)\|_2^2$$

Then

$$0 \le \Delta^{t2}_{m,n}\Delta^{\gamma 2}_{m,n} = \left(\int t^2|\phi(t)|^2\, dt - \left(\int t|\phi(t)|^2\, dt\right)^2\right)\left(\int \gamma^2|\hat{\phi}(\gamma)|^2\, d\gamma - \left(\int \gamma|\hat{\phi}(\gamma)|^2\, d\gamma\right)^2\right).$$

3 Strong uncertainty and complex analysis.

Proposition 2.1 allows us to conclude that

$$\sup_{m,n} \Delta x^2_{m,n}\Delta p^2_{m,n} < \infty.$$

On the other hand, if the expected values $(t_{m,n}, \gamma_{m,n})$ are replaced by a fixed pair (t_0, γ_0), then it is *easy* to see that we have the "strong" uncertainty property,

$$\sup_{m,n} \|(t - t_0)\phi_{m,n}(t)\|_2 \|(\gamma - \gamma_0)\hat{\phi}_{m,n}(\gamma)\|_2 = \infty. \qquad (3.1)$$

The terminology, "weak" and "strong" uncertainty, is discussed in [Ben90] but is not essential to our discussion here. (In fact, at the risk of making an apparently confusing remark, (3.1) is really a weak form of strong uncertainty since it is valid only because of translation from the origin, not because of internal changes within each $\phi_{m,n}$.)

In this section we propose to give a complicated proof of (3.1) for dealing with refinements of (3.1) in which the sup is replaced with various paths in $\mathbf{Z} \times \mathbf{Z}$.

Lemma 3.1. *Suppose f is analytic in the tube $\Re \times (0, b)$, f has nontangential limits a.e. on $\Re$, and $f = 0$ on $K \subseteq \Re$ for which $|K| > 0$. Then f is the zero function.*

In this generality, D. Hamilton has pointed out that topological arguments à la Privalov's theorem can be used in the proof. Alternatively, one could proceed in the usual way by Jensen's theorem after setting up the following mapping. Choose $C \subseteq K$, $|C| > 0$, such that $|f(z)| \le A$ on $\bigcup T_\gamma$, $\gamma \in F$, where $T_\gamma \subseteq \Re \times [0, b)$ is a triangle and $T_\gamma \setminus \{\gamma\} \subseteq \Re \times (0, b)$. For an appropriate set $B \supseteq \bigcup T_\gamma$ choose a bi-absolutely continuous map from B onto the unit disc; and then apply the Jensen's theorem argument found in the books of Koosis (pp. 76–77), Levinson (p. 74), or Porcelli (p. 60).

Lemma 3.2. *Suppose $F \in L^2(\hat{\Re}) \setminus \{0\}$ and supp F is contained in a half-line. Then $\hat{F} = f \in H^2(\Re) \setminus \{0\}$ and, in particular, f is an analytic function with $\Re$ as a boundary.*

Lemma 3.2 is the easy direction of the Paley-Wiener theorem, and also ties in with the work of F. and M. Riesz, Levinson, and Beurling and Malliavin. In light of *Lemma 3.2* and the following application, we note that we are only using *Lemma 3.1* in the case $f \in H^2(\Re)$.

Using *Lemmas 3.1* and *3.2*, our complicated proof of (3.1) proceeds as follows. We first write

$$\begin{aligned} v(m,n) &= \|(t-t_0)\phi_{m,n}(t)\|_2^2 \|(\gamma-\gamma_0)\hat{\phi}_{m,n}(\gamma)\|_2^2 \\ &= \int |(u-t_0+na)\phi(u)|^2\,du \int |(\lambda-\gamma_0+mb)\hat{\phi}(\lambda)|^2\,d\lambda. \end{aligned}$$

For *case 1*, suppose supp ϕ is not contained in a half-line so that for all $n \geq 0$

$$v(m,n) \geq (na)^2 \int_{t_0}^{\infty} |\phi(u)|^2\,du \int |(\lambda-\gamma_0+mb)\hat{\phi}(\lambda)|^2\,d\lambda.$$

Thus,

$$\forall m, \overline{\lim_{n\to\infty}}\, v(m,n) = \infty.$$

For *case 2*, suppose supp ϕ is contained in a half-line. Then for all $m \geq 0$,

$$v(m,n) \geq (mb)^2 \int |(u-t_0+na)\phi(u)|^2\,du \int_{\gamma_0}^{\infty} |\hat{\phi}(\lambda)|^2\,d\lambda.$$

By *Lemmas 3.1* and *3.2*, $\int_{\gamma_0}^{\infty} |\hat{\phi}(\lambda)|^2\,d\lambda > 0$ and so

$$\forall n, \overline{\lim_{m\to\infty}}\, v(m,n) = \infty.$$

3.3 Remark.

(2.1), (3.1), and *Proposition 2.1* can be compared with Bourgain's weak uncertainty inequality [Bou88], cf. [Ben90, section 4]. Bourgain's orthonormal basis was constructed by perturbations of Heisenberg wavelets.

4 The Balian conjecture.

The Balian-Low theorem, mentioned in Section 1, was first stated by Balian [Bal81] in 1981. At the same time he posed the following *conjecture*: there are no orthonormal bases $\{\theta_k\} \subseteq L^2(\Re)$ with the property

$$\sup_k \Delta t_k^2 < \infty \quad \text{and} \quad \sup_k \Delta\gamma_k^2 < \infty,$$

where

$$\begin{aligned} \Delta t_k^2 &= \langle \theta_k(t), t^2\theta_k(t)\rangle - \langle \theta_k(t), t\theta_k(t)\rangle \quad \text{and} \\ \Delta\gamma_k^2 &= \langle \hat{\theta}_k(\gamma), \gamma^2\hat{\theta}_k(\gamma)\rangle - \langle \hat{\theta}_k(\gamma), \gamma\hat{\theta}_k(\gamma)\rangle, \end{aligned}$$

cf. *Proposition 2.1.*

The proof of (2.1) shows the validity of the conjecture for affine wavelets, and the Balian-Low theorem confirms the conjecture for Heisenberg wavelets. Bourgain's theorem casts some doubt on the conjecture, cf. [DJJ].

References.

[Bal81] R. Balian. Un principe d'incertitude fort en théorie du signal ou en mécanique quantique. *C.R. Acad. Sci.*, 292:1357–1362, 1981. Paris.

[Ben89] J. Benedetto. Gabor representations and wavelets. *AMS Contemp. Math.*, 91:9–27, 1989.

[Ben90] J. Benedetto. Uncertainty principle inequalities and spectrum estimation. In J.S. Byrnes and J.L. Byrnes, editors, *Proc. NATO ASI Fourier Analysis and Applications.* Kluwer Academic Publishers, 1990. The Netherlands.

[BHW] J. Benedetto, C. Heil, and D. Walnut. Remarks on the proof of the Balian-Low theorem. To appear.

[Bou88] J. Bourgain. A remark on the uncertainty principle for Hilbertian basis. *J. Functional Analysis*, 79:136–143, 1988.

[CM80] T. Claasen and W. Mecklenbrauker. The Wigner distribution—a tool for time-frequency signal analysis, I–III. *Phillips J. of Research*, 35:217–250, 276–300, 372–389, 1980.

[DJJ] I. Daubechies, S. Jafford, and J.-L. Journé. A simple Wilson orthonormal basis with exponential decay. To appear.

[Gab46] D. Gabor. Theory of communication. *J. of IEE*, 93:429–457, 1946.

[HW89] C. Heil and D. Walnut. Continuous and discrete wavelet transforms. *SIAM Review*, 31:628–666, 1989.

[SW71] E. Stein and G. Weiss. *Introduction to Fourier Analysis on Euclidean Spaces.* Princeton University Press, 1971.

Prometheus Inc.
Final Report
11 July 1990

Section III
A Wiener-Wintner Theorem

A MULTIDIMENSIONAL WIENER-WINTNER THEOREM AND SPECTRUM ESTIMATION

John J. Benedetto
Prometheus Inc.

Abstract

Sufficient conditions are given for a bounded positive measure μ on $\mathbb{R}^d$ to be the power spectrum of a function φ. Applications to spectrum estimation are made for the cases in which a signal φ is known or its autocorrelation P_ϕ is known. In the first case, it is shown that

$$\int |\hat{f}(\gamma)|^2 d\mu_\phi(\gamma) = \lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R)} |f*\varphi(t)|^2 dt,$$

where $\hat{P}_\varphi = \mu_\varphi$, $B(R)$ is the d-dimensional ball of radius R, and f ranges through a prescribed function space. In the second case, an example, which is a variant of the Tomas-Stein restriction theorem, is

$$\forall f \in L^1(\mathbb{R}^d) \cap L^p(\mathbb{R}^d), \quad \left(\int_{\Sigma_{d-1}} |\hat{f}(\theta)|^2 d\mu_{d-1}(\theta) \right)^{1/2} \leq \left[\frac{1}{2} \| \check{\mu}_{d-1} \|_{p'}^{1/2} \right] \left[\|f\|_1 + \|f\|_p \right],$$

where $1 \leq p < 2d/(d+1)$ and the power spectrum μ_{d-1} is the compactly supported restriction of surface measure to the unit sphere $\Sigma_{d-1} \subseteq \hat{\mathbb{R}}^d$.

The author is also Professor of Mathematics at the University of Maryland, College Park. Research sponsored by the Air Force Office of Scientific Research (AFSC), under Contract F49620-88-C-0028. The United States government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

1. Introduction

Wiener's generalized harmonic analysis is a multi-faceted subject. It was conceived to deal with noise, hidden periodicities, and turbulence; it was developed in terms of Tauberian theorems and sophisticated spectral analysis; and it continues to play a role in prediction theory, ergodic theory, stochastic processes, and, of course, harmonic analysis [B;W]. An important fact from the general theory is a theorem due to Wiener and Wintner [WW,241-242]: each bounded positive measure μ on the real-line is the Fourier transform of a function P which can be written in the form,

$$P(t) = \lim_{R\to\infty} \frac{1}{2R} \int_{-R}^{R} \varphi(t+u)\overline{\varphi(u)}du, \tag{1.1}$$

i.e., each μ is the power spectrum of some signal φ having autocorrelation P. In Section 3 we extend this result to functions φ on d-dimensional euclidean space $\mathbb{R}^d$; and we use the theorem as background for proving a d-dimensional type of Wiener-Plancherel formula in Section 5 and a weighted Fourier transform norm inequality and restriction theorem in Section 6. The results in Sections 5 and 6 are viewed in terms of estimating the power spectrum, i.e., "spectrum estimation".

Section 2 is devoted to notation. Theorem 3.3 (in Section 3) is the d-dimensional version of Wiener and Wintner's theorem. It should be pointed out that their proof is cryptic, and our proof on $\mathbb{R}^d$ is inspired by the intricate argument of Bertrandias for $\mathbb{R}$ [Be], cf., [Ba, Chapitre 2]. Section 4 provides a geometrical application of Theorem 3.3 for μ_{d-1}, the restriction of surface measure σ_{d-1} to the unit sphere Σ_{d-1} of d-dimensional euclidean space $\hat{\mathbb{R}}^d(=\mathbb{R}^d)$.

In Theorem 5.2 (of Section 5) we derive a formula to express $\int|\hat{f}(\gamma)|^2 d\mu_\varphi(\gamma)$ in terms of arithmetic means involving f and a given

function φ. (Integration is over $\hat{\mathbb{R}}^d$ and μ_φ is a bounded positive measure on $\hat{\mathbb{R}}^d$ which is the Fourier transform of a d-dimensional version of the function P in (1.1); the function $\hat{f}$ denotes the Fourier transform $\hat{f}(\gamma) = \int f(t)e^{-2\pi i t\cdot\gamma}dt$, where integration is over $\mathbb{R}^d$ and $\gamma \in \hat{\mathbb{R}}^d$.) Using Theorem 5.2 with functions $\hat{f}$ having the appropriate shape, we can estimate the mass of μ_φ on prescribed regions of $\hat{\mathbb{R}}^d$ in terms of the known functions f and φ. This is a form of spectrum estimation and further remarks are made in Section 5.

The results in Section 6 are elementary but of some interest because of recent work on weighted norm inequalities and restriction theorems. The (L^2, L^p) restriction theorem of Tomas and Stein asserts that

$$(1.2) \qquad \forall f \in L^1(\mathbb{R}^d)\cap L^p(\mathbb{R}^d), \quad \left(\int_{\Sigma_{d-1}} |\hat{f}(\theta)|^2 d\sigma_{d-1}(\theta)\right)^{1/2} \le c(p)\|f\|_p$$

for any fixed $1 \le p \le 2(d+1)/(d+3)$, e.g., [T]. $L^p(\mathbb{R}^d)$ and $\|f\|_p$ are the usual Lebesgue space and norm. Discussions of the role of curvature and extensions of (1.2) to other smooth sub-manifolds are found in [CD, 103-109; S, 325-329]. Weighted L^p versions of (1.2) are found in [BH]. Proposition 6.1 is a μ-weighted Fourier transform norm inequality with explicit norm constant in terms of the inverse Fourier transform P of μ; and Proposition 6.2 is a corollary which can be viewed as a restriction result since we take $\mu = \mu_{d-1}$. The inequality (1.2) is deep and Proposition 6.2 is elementary. In (1.2) the constant c(p) is not explicit, $p = 2(d+1)/(d+3)$ is largest possible, and the right hand norm is $\|f\|_p$. In Proposition 6.2 the constant is explicit and the values of p extend beyond $2(d+1)/(d+3)$, but the right hand norm is $\|f\|_1 + \|f\|_p$. As far as spectrum estimation is concerned, Proposition 6.2 can be used to give an upper bound of the mass of μ_{d-1} on prescribed regions of $\hat{\mathbb{R}}^d$ in terms of the known functions f and P. This,

too, is a form of spectrum estimation and further remarks are made in Section 6. Naturally, in light of Theorem 3.3 and the generality of Proposition 6.1, our discussion of μ_{d-1} extends to many other measures.

Acknowledgement

I would like to acknowledge important observations about the contents of this paper by Sadahiro Saeki, David Walnut, and Elmar Winkelnkemper, as well as an overall discussion with Ward Evans, Tom Harrison, Christopher Heil, and Rodney Kerby.

2. Notation

Besides the L^p-spaces, $1 \le p < \infty$, mentioned in Section 1, we shall deal with a number of other spaces. To introduce them we let $\hat{\mathbb{R}}^d (= \mathbb{R}^d)$ be the dual group of d-dimensional euclidean space $\mathbb{R}^d$ and let X be a locally compact subspace of $\hat{\mathbb{R}}^d$. Then $C_c(X)$ is the vector space of complex-valued continuous functions $f : X \longrightarrow \mathbb{C}$ having compact support $\operatorname{supp} f \subseteq X$. A measure μ on X is a linear functional defined on $C_c(X)$ satisfying $\lim_{j\to\infty} \langle \mu, f_j \rangle = 0$ for every sequence $\{f_j\} \subseteq C_c(X)$ having the properties that $\lim_{j\to\infty} \|f_j\|_\infty = 0$ and $\operatorname{supp} f_j \subseteq K$, where $K \subseteq X$ is a compact set independent of j and $\|\cdots\|_\infty$ is the usual sup norm, e.g., [Bo]. $M(X)$ is the space of measures on X and $M_+(X) = \{\mu \in M(X) : \langle \mu, f \rangle \ge 0$ for all non-negative $f \in C_c(X)\}$ is the space of positive measures on X. Similarly, $M_b(X)$ is the subspace of $M(X)$ having bounded variation, i.e., the above mentioned convergence criterion on $C_c(X)$ is replaced by $(C_c(X), \|\cdots\|_\infty)$; and $M_{b+}(X)$ consists of the positive elements of $M_b(X)$. The support of $\mu \in M(X)$ is denoted by $\operatorname{supp} \mu$. We write $\langle \mu, f \rangle = \int_X f(\gamma) d\mu(\gamma)$ and in the

case $X = \hat{\mathbb{R}}^d$ we write $\langle \mu, f \rangle = \int f(\gamma) d\mu(\gamma)$. If $\mu \in M_b(X)$ then μ is well defined on $C_b(X) = \{f : f$ is continuous and bounded on $X\}$.

For $p \in (0,\infty)$, $L^p_{loc}(\mathbb{R}^d)$ is the set of functions $f : \mathbb{R}^d \to \mathbb{C}$ for which $|f|^p$ is locally integrable with respect to Lebesgue measure. If $\mu \in M_+(\hat{\mathbb{R}}^d)$ then $L^p_\mu(\hat{\mathbb{R}}^d)$ designates the set of Borel measurable functions f defined μ a.e. on $\hat{\mathbb{R}}^d$ for which $\|f\|_{p,\mu} = (\int |f(\gamma)|^p d\mu(\gamma))^{1/p} < \infty$. There is the standard adjustment for $p = \infty$. Also, we write $p' = p/(p-1)$.

The Fourier transform defined in Section 1 extends from $L^1(\mathbb{R}^d)$ to $M_b(\mathbb{R}^d)$ (and beyond), and $\mu^\vee$ is the inverse Fourier transform of $\mu \in M_b(\hat{\mathbb{R}}^d)$.

The characteristic function of the set $S \subseteq \mathbb{R}^d$ is χ_S and the Lebesgue measure of S is denoted by $|S|$. We set $B(R) = \{ x \in \mathbb{R}^d : |x| \le R\}$ so that $|B(R)| = \omega_{d-1} R^d/d$ where $\omega_{d-1} = 2\pi^{d/2}/(d\Gamma(d/2))$ is the surface area of Σ_{d-1}. Finally, we mention that we shall frequently deal with sets of the form $B(R)\backslash B(r)$, and that the boundaries of such sets are unimportant for our results. As such, on this point, we shall not usually be concerned about boundaries of such sets in our calculations.

3. The Wiener-Wintner Theorem on $\mathbb{R}^d$

Given $\mu \in M_{b+}(\hat{\mathbb{R}}^d)$ and let δ_ω be the Dirac measure supported by $\{\omega\}$. It is well-known that there is a sequence $\{\mu_n\} \subseteq M_{b+}(\text{supp } \mu)$ of positive discrete measures,

$$\mu_n = \sum_{j=1}^{N_n} a_{j,n} \delta_{\omega_{j,n}}, \quad a_{j,n} > 0,$$

such that $\{\omega_{j,n} : j = 1, \cdots, N_n\} \subseteq \text{supp } \mu$ for each n,

(3.1) $$\lim_{n\to\infty} \langle \mu_n, 1\rangle = \langle \mu, 1\rangle,$$

and $\lim_{n\to\infty} \mu_n = \mu$ in the (vague) topology $\sigma(M_b(\hat{\mathbb{R}}^d), C_c(\hat{\mathbb{R}}^d))$, e.g., [Bo, Chapitre III, Section 2, no. 4]. Actually, (3.1) and the $\sigma(M_b(\hat{\mathbb{R}}^d), C_c(\hat{\mathbb{R}}^d))$ convergence allow us to conclude that $\lim_{n\to\infty} \mu_n = \mu$ in the "Levy" topology $\sigma(M_b(\hat{\mathbb{R}}^d), C_b(\hat{\mathbb{R}}^d))$, e.g., [Ma, 91-93].

For a given $\mu \in M_{b+}(\hat{\mathbb{R}}^d)$ and sequence $\{\mu_n\} \subseteq M_{b+}(\text{supp } \mu)$ as above, we define

$$\varphi_n(t) = \sum_{j=1}^{N_n} a_{j,n}^{1/2} e^{2\pi i t\cdot\omega_{j,n}}$$

so that

$$\|\varphi_n\|_\infty \le \sum_{j=1}^{N_n} a_{j,n}^{1/2} \le N_n \sup_{1\le j\le N_n} a_{j,n}^{1/2}.$$

Lemma 3.1. For each n,

(3.2) $$\lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R)} \varphi_n(t+x)\overline{\varphi_n(x)}dx = \check{\mu_n}(t),$$

uniformly on $\mathbb{R}^d$.

Proof. The left hand side of (3.2) is $\check{\mu_n}(t) + \lim_{R\to\infty} \varepsilon(R,t,n)$ (if this limit exists), where

$$\varepsilon(R,t,n) = \frac{1}{|B(R)|} \sum_{j\neq k} a_{j,n}^{1/2} a_{k,n}^{1/2} e^{2\pi i t\cdot\omega_{j,n}} \int_{B(R)} e^{2\pi i x\cdot(\omega_{j,n}-\omega_{k,n})}dx$$

$$= \frac{2\pi}{|B(R)|} \sum_{j\neq k} a_{j,n}^{1/2} a_{k,n}^{1/2} e^{2\pi i t\cdot\omega_{j,n}} |\omega_{j,n}-\omega_{k,n}|^{-(\frac{d-2}{2})} \int_0^R r^{\frac{d}{2}} J_{\frac{d-2}{2}}(2\pi r|\omega_{j,n}-\omega_{k,n}|)dr$$

and $J_{\frac{d-2}{2}}$ is the Bessel function of order $(d-2)/2$. Since $|J_{\frac{d-2}{2}}(s)| \le 1$ (at the origin) and $J_{\frac{d-2}{2}}(s) = O(s^{-1/2})$, $s \to \infty$, we see that $\lim_{R\to\infty} \varepsilon(R,t,n) = 0$ independently of t.

q.e.d.

We shall also need the following easy fact.

Lemma 3.2. Given a sequence $\{R_n\}$ increasing to infinity and $\varphi \in L^\infty_{loc}(\mathbb{R}^d)$. Fix $t \in \mathbb{R}^d$. Then for each fixed p,

$$\lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R_p)} \varphi(t+x)\overline{\varphi(x)}dx = 0.$$

We use Lemma 3.1 to define a specific sequence $\{R_n\}$ and a specific function φ in the following way. From the uniform convergence we know that

$$\forall n \ge 1,\ \exists A_n \ge A_{n-1} \text{ such that } \forall t \in \mathbb{R}^d \text{ and } \forall R \ge A_n,$$

(3.3)

$$\left| \frac{1}{|B(R)|} \int_{B(R)} \varphi_n(t+x)\overline{\varphi_n(x)}dx - \check{\mu_n}(t) \right| < \frac{1}{2^{n+1}}.$$

We set $R_n = (A_1+1)(A_2+2)\cdots(A_n+n)$ so that each $R_n \ge n!$ and the sequences $\{R_n\}$, $\{R_{n+1}/R_n\} = \{A_{n+1}+n+1\}$, and $\{R_{n+1}-R_n\} = \{(A_1+1)\cdots(A_n+n)(A_{n+1}+n)\}$ increase to infinity. For this sequence $\{R_n\}$ and for $\{\varphi_n\}$ defined above we define φ on $\mathbb{R}^d$ by setting $\varphi(t) = \varphi_n(t)$ for $R_n < |t| < R_{n+1}$, $n \ge 1$, and letting $\varphi(t) = 0$ for $|t| < R_1$. Clearly, $\varphi \in L^\infty_{loc}(\mathbb{R}^d)$ and the values of $\varphi(t)$ for $|t| = r_n$ are not important.

Using this notation, setup, and the lemmas, we can state the Wiener-Wintner construction on $\mathbb{R}^d$.

Theorem 3.3. Given $\mu \in M_{b+}(\hat{\mathbb{R}}^d)$ with corresponding functions $\{\varphi_n\}$ and $\varphi \in L^\infty_{loc}(\mathbb{R}^d)$. Assume

$$\lim_{n\to\infty} \frac{1}{n!} \sum_{j=2}^{n} \max(\|\varphi_{j-1}\|_\infty, \|\varphi_j\|_\infty)\max(\|\varphi_j\|_\infty, \|\varphi_{j+1}\|_\infty) = 0. \tag{3.4}$$

Then, for each $t \in \mathbb{R}^d$,

$$\lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R)} \varphi(t+x)\overline{\varphi(x)}dx = \check{\mu}(t). \tag{3.5}$$

Proof. a. Given $t \in \mathbb{R}^d$ and $\varepsilon > 0$. Because of our setup we can choose $p_1 = p_1(t)$ for which $R_{k+1} - |t| > R_k + |t|$ for all $k \geq p_1$ and we can choose $p_2 = p_2(\varepsilon, t) > p_1$ such that

$$\forall k \geq p_2, \quad |\check{\mu_k}(t) - \check{\mu}(t)| < \varepsilon. \tag{3.6}$$

We shall find $p = p(\varepsilon, t) > p_2$ such that for all sufficiently large R,

$$\left|\frac{1}{|B(R)|} \int_{B(R)\setminus B(R_p)} \varphi(t+x)\overline{\varphi(x)}dx - \check{\mu}(t)\right| < 14\varepsilon; \tag{3.7}$$

and, hence, (3.5) is a consequence of Lemma 3.2 and a "$\overline{\lim}$" argument.

b. For each $p > p_2$ and each $R > R_p$, write

$$\begin{aligned}\frac{1}{|B(R)|} \int_{B(R)\setminus B(R_p)} \varphi(t+x)\overline{\varphi(x)}dx &= \sum_{k=p}^{n-1} (b(k,R) + c(k,R)) \\ &+ \frac{1}{|B(R)|} \int_{B(R)\setminus B(R_n)} \varphi(t+x)\overline{\varphi(x)}dx,\end{aligned} \tag{3.8}$$

where

$$b(k,R) = \frac{1}{|B(R)|} \int_{B(R_{k+1}-|t|)\setminus B(R_k)} \varphi(t+x)\overline{\varphi(x)}dx,$$

$$c(k,R) = \frac{1}{|B(R)|} \int_{B(R_{k+1})\setminus B(R_{k+1}-|t|)} \varphi(t+x)\overline{\varphi(x)}dx,$$

and $n = n(R)$ is the largest integer n for which $R_n \le R$.

In part c we shall verify that for $p > p_1$

$$\lim_{R\to\infty} \sum_{k=p}^{n-1} c(k,R) = 0. \tag{3.9}$$

Parts d - h are devoted to showing that there are $p_3 = p_3(\varepsilon,t) > p_2$ and $R(\varepsilon,t) > p_3$ such that, for $R(\varepsilon,t) > p \ge p_3$,

$$\forall R > R(\varepsilon,t), \quad \sum_{k=p}^{n-1} b(k,R)$$

$$= \check{\mu}(t) \frac{1}{|B(R)|} \sum_{k=p}^{n-1} (|B(R_{k+1}-|t|)| - |B(R_k)|) + r(p,R,t), \tag{3.10}$$

where $|r(p,R,t)| < 7\varepsilon + \frac{1}{2^{p-1}}$. Parts i - k contain the proof that

$$\forall R > R(\varepsilon,t), \quad \frac{1}{|B(R)|} \int_{B(R)\setminus B(R_n)} \varphi(t+x)\overline{\varphi(x)}dx = \check{\mu}(t)\left(1 - \frac{|B(R_n)|}{|B(R)|}\right) + s(R,t), \tag{3.11}$$

where $|s(R,t)| < 4\varepsilon + \frac{1}{2^n}$. (3.7) is obtained in part l by combining (3.9), (3.10), and (3.11).

c. If $x \in B(R_{k+1})\setminus B(R_{k+1} - |t|)$ then $R_k \le |x+t| \le R_{k+2}$. Thus, setting $c_k = \|\varphi_k\|_\infty \max(\|\varphi_k\|_\infty, \|\varphi_{k+1}\|_\infty)$, we have

$$\sum_{k=p}^{n-1} c(k,R) \le \frac{1}{R^d} \sum_{k=p}^{n-1} c_k \left[R_{k+1}^d - (R_{k+1}-|t|)^d \right]$$

$$\le \frac{1}{R_n^d} \sum_{k=p}^{n-1} c_k \, R_{k+1}^d \left[1 - (1-\frac{|t|}{R_{k+1}})^d \right]$$

$$\le \frac{1}{R_n^d} \sum_{k=p}^{n-1} c_k \, R_{k+1}^d \sum_{j=1}^{d} \binom{d}{j} \left(\frac{|t|}{R_{k+1}} \right)^j$$

$$\le \left(\frac{1+|t|}{R_n} \right)^d \sum_{k=p}^{n-1} c_k \, R_{k+1}^{d-1} \le \frac{(1+|t|)^d}{n!} \sum_{k=1}^{n-1} c_k .$$

(3.9) follows by comparing the right hand side of this estimate with our hypothesis (3.4).

d. In order to estimate $b(k,R)$ we define the sets $A_k(t) = \{x \in B(R_{k+1}-|t|)\setminus B(R_k) : t + x \in B(R_{k+1})\setminus B(R_k)\}$ and $A_{k-1}(t) = \{x \in B(R_{k+1}-|t|)\setminus B(R_k) : t + x \in B(R_k)\setminus B(R_{k-1})\}$. Then, for $k > p_1$, $B(R_{k+1}-|t|)\setminus B(R_k) = A_k(t) \cup A_{k-1}(t)$, a disjoint union; and we have

$$b(k,R) = \frac{1}{|B(R)|} \int_{A_k(t)\cup B(R_k)} \varphi_k(t+x)\overline{\varphi_k(x)}dx$$

(3.12)

$$- \frac{1}{|B(R)|} \int_{B(R_k)} \varphi_k(t+x)\overline{\varphi_k(x)}dx + \frac{1}{|B(R)|} \int_{A_{k-1}(t)} \varphi_{k-1}(t+x)\overline{\varphi_k(x)}dx.$$

We shall estimate these three integrals. The first two will involve the factor $\check{\mu}(t)$ and the latter becomes small by the size of $A_{k-1}(t)$.

In the process of making these estimates we need the bound,

$$\frac{1}{R_n^d} \sum_{k=1}^{n} R_k^d \le 2. \tag{3.13}$$

In fact, the left hand side of (3.13) is bounded by

$$1 + \frac{1}{n} + \frac{1}{n(n-1)} + \cdots + \frac{1}{n!},$$

and it is easy to see that for each $k \geq 3$

$$\frac{1}{(k+1)!}\sum_{j=0}^{k+1} j! < \frac{1}{k!}\sum_{j=0}^{k} j! \leq \frac{5}{3};$$

hence, (3.13) is valid.

e. Our initial step in estimating the first integral on the right hand side of (3.12) is to prove that there is $p_3 = p_3(\varepsilon,t) > p_2$ such that for all $k \geq p_3$,

(3.14)

$$\left|\frac{1}{|B(R_{k+1}-|t|)|}\left[\int_{B(R_{k+1}-|t|)} \varphi_k(t+x)\overline{\varphi_k(x)}dx - \int_{A_k(t)\cup B(R_k)} \varphi_k(t+x)\overline{\varphi_k(x)}dx\right]\right| < \varepsilon.$$

The difference of the two integrals is the integral over $A_{k-1}(t)$, and if $x \in A_{k-1}(t)$ then $R_k \leq |x| \leq R_k + |t|$. Thus, setting $d_k = \|\varphi_k\|_\infty^2$, the left hand side of (3.14) is bounded by

$$(3.15) \qquad \frac{d_k}{(R_{k+1}-|t|)^d}\left[(R_k+|t|)^d - R_k^d\right] \leq \frac{d\, d_k}{(R_{k+1}-|t|)^d}(R_k+|t|)^{d-1},$$

where this last inequality is a consequence of the mean value theorem. By (3.4), the right hand side of (3.15) tends to 0 as $k \longrightarrow \infty$; and (3.14) is obtained.

By means of (3.3), (3.6), (3.14), and the triangle inequality we have

$$(3.16) \qquad \left|\frac{1}{|B(R_{k+1}-|t|)|}\int_{A_k(t)\cup B(R_k)} \varphi_k(t+x)\overline{\varphi_k(x)}dx - \check{\mu}(t)\right| < 2\varepsilon + \frac{1}{2^{k+1}},$$

where $k \geq p_3$. Clearly, we can write (3.16) in the form,

$$(3.17)\qquad \frac{1}{|B(R_{k+1}-|t|)|}\int_{A_k(t)\cup B(R_k)}\varphi_k(t+x)\overline{\varphi_k(x)}dx = \mu^{\vee}(t) + \beta_k(t)(2\varepsilon + \frac{1}{2^{k+1}}),$$

where $|\beta_k(t)| < 1$ and $k \geq p_3$.

f. To estimate the second integral on the right hand side of (3.12) we use (3.3), (3.6), and the triangle inequality to write

$$(3.18)\qquad \left|\frac{1}{B(R_k)}\int_{B(R_k)}\varphi_k(t+x)\overline{\varphi_k(x)}dx - \mu^{\vee}(t)\right| < \varepsilon + \frac{1}{2^{k+1}},$$

for $k > p_2$. Analogous to part e, (3.18) yields

$$(3.19)\qquad \frac{1}{|B(R_k)|}\int_{B(R_k)}\varphi_k(t+x)\overline{\varphi_k(x)}dx = \mu^{\vee}(t) + \gamma_k(t)(\varepsilon + \frac{1}{2^{k+1}}),$$

where $|\gamma_k(t)| < 1$ and $k > p_2$.

g. Since $A_{k-1}(t) \subseteq B(R_k+|t|)\setminus B(R_k)$, the third integral on the right hand side of (3.12) is bounded by

$$(3.20)\qquad \frac{1}{|B(R_k)|}\int_{A_{k-1}(t)}|\varphi_{k-1}(t+x)\overline{\varphi_k(x)}|dx \leq \frac{\|\varphi_{k-1}\|_\infty\|\varphi_k\|_\infty}{R_n^d}\left[(R_k+|t|)^d - R_k^d\right].$$

h. By parts e, f, and g we can write

$$\begin{aligned}\sum_{k=p}^{n-1} b(k,R) &= \frac{1}{|B(R)|}\sum_{k=p}^{n-1}|B(R_{k+1}-|t|)|(\mu^{\vee}(t) + \beta_k(t)(2\varepsilon + \frac{1}{2^{k+1}}))\\ &\quad - \frac{1}{|B(R)|}\sum_{k=p}^{n-1}(|B(R_k)|)(\mu^{\vee}(t) + \gamma_k(t)(\varepsilon + \frac{1}{2^{k+1}}))\\ &\quad + \frac{1}{|B(R)|}\sum_{k=p}^{n-1}\int_{A_{k-1}(t)}\varphi_{k-1}(t+x)\overline{\varphi_k(x)}dx.\end{aligned}\qquad (3.21)$$

To estimate the last term in (3.21) we use (3.20) and compute

$$\frac{1}{R_n^d}\sum_{k=p}^{n-1}\|\varphi_{k-1}\|_\infty\|\varphi_k\|_\infty\left[(R_k+|t|)^d - R_k^d\right]$$

$$\leq \frac{d|t|}{R_n^d}\sum_{k=p}^{n-1}\|\varphi_{k-1}\|_\infty\|\varphi_k\|_\infty\,(R_k+|t|)^{d-1}$$

$$\leq \frac{d|t|2^{d-1}}{R_n}\sum_{k=p}^{n-1}\|\varphi_{k-1}\|_\infty\|\varphi_k\|_\infty\,(\frac{R_k}{R_n})^{d-1}.$$

By (3.4), this term is less than ε for all large R. Consequently, (3.21) allows us to obtain (3.10). It is at this point that we invoke (3.13).

i. We now estimate the last term in (3.8). Analogous to part d we define the sets $A_{R,k}(t) = \{x \in B(R)\backslash B(R_n) : t + x \in B(R_{k+1})\backslash B(R_k)\}$ for $k = n-1,\ n,\ n+1$. Then, for $n > p_1$, $B(R)\backslash B(R_n) = A_{R,n-1}(t)\cup A_{R,n}(t)\cup A_{R,n+1}(t)$, a disjoint union. (Actually, we have $n \geq p_3$.) Therefore, the last term of (3.8) is

$$\frac{1}{|B(R)|}\int_{B(R)\backslash B(R_n)}\varphi(t+x)\overline{\varphi(x)}dx = \frac{1}{|B(R)|}\int_{A_{R,n}(t)\cup B(R_n)}\varphi_n(t+x)\overline{\varphi_n(x)}dx,$$

$$(3.22)\qquad - \frac{|B(R_n)|}{|B(R)|}\left[\frac{1}{|B(R_n)|}\int_{B(R_n)}\varphi_n(t+x)\overline{\varphi_n(x)}dx\right]$$

$$+ \frac{1}{|B(R)|}\left[\int_{A_{R,n-1}(t)}\varphi_{n-1}(t+x)\overline{\varphi_n(x)}dx + \int_{A_{R,n+1}(t)}\varphi_{n+1}(t+x)\overline{\varphi_n(x)}dx\right].$$

j. The last term of (3.22) is estimated as in part g. The inclusions $A_{R,n-1}(t) \subseteq B(R_n+|t|)\backslash B(R_n)$ and $A_{R,n+1}(t) \subseteq B(R)\backslash B(R_{n+1}-|t|)$ are valid, but it should be observed that $A_{R,n+1}(t)$ can be the empty set, e.g., if

$R < R_{n+1} - |t|$. In any case, this last term of (3.22) is dominated by

$$(3.23)\quad \frac{1}{R^d}\left\{\|\varphi_{n-1}\|_\infty\|\varphi_n\|_\infty\left[(R_n+|t|)^d - R_n^d\right] + \|\varphi_{n+1}\|_\infty\|\varphi_n\|_\infty\left[R^d - (R_{n+1}-|t|)^d\right]\right\},$$

where "$[R^d - (R_n-|t|)^d]$" is defined to be 0 if $R < R_{n+1} - |t|$. Using the mean value theorem again, (3.23) is bounded by

$$\frac{d|t|}{R_n}\|\varphi_{n-1}\|_\infty\|\varphi_n\|_\infty\left(\frac{R_n+|t|}{R_n}\right)^{d-1} + \frac{d(R-(R_{n+1}-|t|))}{R}\|\varphi_{n+1}\|_\infty\|\varphi_n\|_\infty.$$

Our hypothesis (3.4) allows us to conclude that this quantity is less than ε for all large R. (For the second term, recall that $R < R_{n+1}$.)

k. Our estimation of the first two integrals on the right hand side of (3.22) follows steps e and f. For example, analogous to (3.14), we have

$$\left|\frac{1}{|B(R)|}\left[\int_{B(R)}\varphi_n(t+x)\overline{\varphi_n(x)}dx - \int_{A_{R,n}(t)\cup B(R_n)}\varphi_n(t+x)\overline{\varphi_n(x)}dx\right]\right|$$

$$\le \frac{1}{|B(R)|}\|\varphi_n\|_\infty^2\left[|A_{R,n-1}(t)| + |A_{R,n+1}(t)|\right];$$

and the right hand side is less than ε for all large R by the calculation used in step j. Proceeding in this way, as in parts e and f, we obtain

$$\frac{1}{|B(R)|}\int_{A_{R,n}(t)\cup B(R_n)}\varphi_n(t+x)\overline{\varphi_n(x)}dx = \check{\mu}(t) + \beta_{R,n}(t)(2\varepsilon + \frac{1}{2^{n+1}})$$

and

$$\frac{1}{|B(R_n)|}\int_{B(R_n)}\varphi_n(t+x)\overline{\varphi_n(x)}dx = \check{\mu}(t) + \gamma_{R,n}(t)(\varepsilon + \frac{1}{2^{n+1}}),$$

where $|\beta_{R,n}(t)|$, $|\gamma_{R,n}(t)| < 1$. Combining these facts with part j and (3.22), we have verified (3.11).

1. For $p \geq p_3$ large enough and for all large R, $n(R) > p$, the right hand side of (3.8) is

$$\check{\mu}(t) + \frac{\check{\mu}(t)}{|B(R)|}\left[\sum_{k=p}^{n-1}\left(|B(R_{k+1}-|t|)| - |B(R_k)|\right) - |B(R_n)|\right] \tag{3.24}$$

plus an error term bounded by 13ε. This is a consequence of (3.9), (3.10), and (3.11). We rewrite the second term of (3.24) as

$$-\frac{\check{\mu}(t)}{|B(R)|}\left[\left(|B(R_n)| - |B(R_n-|t|)|\right) + \left(|B(R_{n-1})| - |B(R_{n-1}-|t|)|\right)\right.$$
$$\left. +\cdots+ \left(|B(R_{p+1})| - |B(R_{p+1}-|t|)|\right) + |B(R_p)|\right],$$

which, in absolute value, is bounded by

$$\frac{|\check{\mu}(t)|}{R^d}\left[\sum_{k=p}^{n}\left((R_k^d - (R_k-|t|)^d\right) + R_p^d\right] \leq |\check{\mu}(t)|\frac{d|t|}{R}\sum_{k=p}^{n-1}\left(\frac{R_k}{R}\right)^{d-1} + |\check{\mu}(t)|\left(\frac{R_p}{R}\right)^d,$$

and which, in turn, is less than ε for all large R. Thus (3.7) is verified, and, by the observation in part a, the theorem is proved.

q.e.d.

Remark 3.4. The hypothesis (3.4) can be weakened. For instance, $\|\varphi_k\|_\infty^2 \leq CR_k$ is sufficient. There is also left unanswered the problems of constructing φ for arbitrary $\mu \in M_{b+}(\hat{\mathbb{R}}^d)$ and for characterizing those μ for which $\varphi \in L^\infty(\mathbb{R}^d)$. The former problem is answered positively for all μ in case $d = 1$; the latter is not yet solved in case $d = 1$.

4. Example

Let $\mu_{d-1}(\gamma) = \delta(|\gamma| - 1) \in M_{b+}(\hat{\mathbb{R}}^d)$ be the measure corresponding to a mass distributed homogeneously over Σ_{d-1}; the total mass is taken as ω_{d-1} so that the compactly supported measure μ_{d-1} is the restriction of surface measure σ_{d-1} to Σ_{d-1}. The transform $\check{\mu}_{d-1}(t)$ is easily computed in terms of the approximants $\frac{1}{\tau}\chi(\frac{|\gamma|-1}{\tau})$ to μ_{d-1}, where $\chi(r) = \chi_{[-1/2,\,1/2)}(r)$ for $r \in \mathbb{R}$. In fact, we have

$$\check{\mu}_{d-1}(t) = \lim_{\tau\to 0} \int \frac{1}{\tau}\chi(\frac{|\gamma|-1}{\tau})\, e^{2\pi i t\cdot\gamma}\, d\gamma$$

(4.1)

$$= \lim_{\tau\to 0} \frac{1}{\tau} \int_{1-\frac{\tau}{2}}^{1+\frac{\tau}{2}} \rho^{d-1} \left(\int_{\Sigma_{d-1}} e^{2\pi i t\cdot\rho\theta}\, d\sigma_{d-1}(\theta)\right) d\rho$$

$$= 2\pi|t|^{-(\frac{d-2}{2})}\, J_{\frac{d-2}{2}}(2\pi|t|),$$

since $\int_{\Sigma_{d-1}} e^{2\pi i t\cdot\rho\theta}\, d\sigma_{d-1}(\theta) = 2\pi(|t|\rho)^{-(\frac{d-2}{2})}\, J_{\frac{d-2}{2}}(2\pi|t|\rho).$

Theorem 4.1. Given $d \geq 2$ and the measure μ_{d-1}. There is $\varphi \in L^2_{loc}(\mathbb{R}^d)$ such that

$$\forall t \in \mathbb{R}^d, \quad \lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R)} \varphi(t+x)\overline{\varphi(x)}dx = \check{\mu}_{d-1}(t).$$

Proof. The result will follow from Theorem 3.3 once we construct $\{\mu_n\}$ and $\{\varphi_n\}$ as in Section 3 where $\{\varphi_n\}$ satisfies (3.4).

To this end we first cover Σ_{d-1} in the following way. For each n we choose about n^{d-1} balls $B_{j,n}$, $j = 1,\cdots,N_n \le N_d\, n^{d-1}$, each having diameter bounded by D_d/n and whose union covers Σ_{d-1}. For $d = 2$, this is doable since the circle can be divided into n arcs of equal length $2\pi/n$, and, hence, n balls of diameter D_2/n can be chosen. For $d = 3$, we first consider the equator and the polar great circle. We choose n equi-spaced points on both circles. For convenience let n be even so that the polar points are marked on the polar great circle. Then spin the polar great circle stopping at each marked point on the equator. At each stop we mark points on Σ_2 determined by the rotated points of the polar great circle. In this way we obtain $N_n \le N_3 n^2$ points on Σ_2. For any marked point p on the equator we choose a ball $B(p,r)$ where the radius r is large enough to include two marked points of the equator on either side of p. Balls of this size centered at the other marked points on Σ_2 yield the desired set $\{B_{j,n}\}$ (since $B(p,r)$ covers the largest area on Σ_2 between a block of 6 neighboring points). For $d > 3$ we use this process by considering $d - 1$ "great circles" Σ_{d-2} to find the n^{d-1} points and isometric balls $B_{j,n}$ centered at these points. Analytically, this can be accomplished by imbedding Σ_{d-1} in the unit cube, covering the surface of the unit cube by $2^d n^{d-1}$ balls of diameter D_d/n (equidistributed on each face of the cube), and linearly sliding each ball to the origin where it is designated $B_{j,n}$ when its center intersects Σ_{d-1}. Clearly (geometrically), these $B_{j,n}$ cover Σ_{d-1}.

We can now construct $\{\mu_n\}$. For each fixed n we set $B_{j,n}(\Sigma_{d-1}) = B_{j,n} \cap \Sigma_{d-1}$, $j = 1,\cdots,N_n$; we then obtain a partition of Σ_{d-1} by defining $O_{1,n} = B_{1,n}(\Sigma_{d-1})$ and

$$O_{j,n} = B_{j,n}(\Sigma_{d-1}) \bigcap \left(\sum_{k=1}^{j-1} B_{k,n}(\Sigma_{d-1})\right)^{\sim}, \quad j = 2, \cdots, N_n.$$

There is K_d, independent of n and $j = 1, \cdots N_n$, such that

$$\mu_{d-1}(O_{j,n}) \leq \frac{K_d}{n^{d-1}}. \tag{4.2}$$

Besides the geometric proof of (4.2) we can proceed analytically as follows. Take $g = \chi_{B_{j,n}(\Sigma_{d-1})}$ and define g_d on $\mathbb{R}^d$ by setting $g_d(0) = 0$, $g_d(\rho\theta) = g(\theta)$ for $0 < \rho \leq 1$ and $\theta \in \Sigma_{d-1}$, and $g_d(\gamma) = 0$ for $|\gamma| > 1$. By the definition of μ_{d-1},

$$\int_{\Sigma_{d-1}} g(\theta) d\sigma_{d-1}(\theta) = \int g_d(\gamma) d\gamma; \tag{4.3}$$

and (4.2) is obtained by calculating the right hand side of (4.3) using spherical coordinates.

Now that we have a partition $\{O_{j,n} : j = 1, \cdots, N_n \leq N_d n^{d-1}\}$ of Σ_{d-1} satisfying (4.2) and having diameters bounded by D_d/n, we choose $\omega_{j,n} \in O_{j,n}$ (when $O_{j,n} \neq \phi$) and define

$$\mu_n = \sum_{j=1}^{N_n} \mu_{d-1}(O_{j,n}) \, \delta_{\omega_{j,n}}$$

and

$$\varphi_n(t) = \sum_{j=1}^{N_n} \mu_{d-1}(O_{j,n})^{1/2} \, e^{2\pi i t \cdot \omega_{j,n}}.$$

It remains to prove that $\lim_{n\to\infty} \mu_n = \mu$ in the Levy topology (so that we have (3.6)) and that (3.4) is satisfied.

The proof that $\lim_{n\to\infty} \mu_n = \mu$ in the Levy topology, i.e., $\sigma(M_b(\hat{\mathbb{R}}^d), C_b(\hat{\mathbb{R}}^d))$, proceeds as follows. Given $g \in C_b(\hat{\mathbb{R}}^d)$; since g is uniformly continuous on Σ_{d-1} there is $\{r_n\}$ decreasing to 0 so that if $\theta,\theta' \in \Sigma_{d-1}$ have Euclidean distance less than D_d/n then $|g(\theta) - g(\theta')| < r_n$. Therefore, for each j,

$$\int_{\Sigma_{d-1}} \chi_{O_{j,n}}(\theta) g(\theta) d\sigma_{d-1}(\theta) = g(\omega_{j,n}) \int_{\Sigma_{d-1}} \chi_{O_{j,n}}(\theta) d\sigma_{d-1}(\theta)$$

$$+ \beta_{j,n}\, r_n\, |\mu_{d-1}|(O_{j,n}),$$

$|\beta_{j,n}| < 1$. Summing over j we obtain the desired convergence since

$$\left|\int g(\gamma) d\mu_{d-1}(\gamma) - \int g(\gamma) d\mu_n(\gamma)\right| < \omega_{d-1}\, r_n.$$

To verify (3.4) we first use (4.2) to observe that

$$\|\varphi_k\|_\infty \le N_d\, K_d^{1/2} k^{\left(\frac{d-1}{2}\right)}.$$

Thus, we have

$$\frac{1}{n!} \sum_{k=2}^{n} \max(\|\varphi_{k-1}\|_\infty, \|\varphi_k\|_\infty) \max(\|\varphi_k\|_\infty, \|\varphi_{k+1}\|_\infty) \le \frac{N_d^2 K_d}{n!} \sum_{k=1}^{n+1} k^{d-1} \le N_d^2 K_d \frac{(n+1)^d}{n!}.$$

The right hand side tends to 0 and so (3.4) is satisfied.

q.e.d.

Remark 4.2. The geometrical construction in Theorem 4.1 was made with (3.4) in mind. There are other methods of partitioning Σ_{d-1} so that the results of Section 3 can be invoked. For example, let $d = 3$ and consider the

"simplex" determined by the six points $(\pm 1,0,0)$, $(0,\pm 1,0)$, and $(0,0,\pm 1)$. The trace of the boundary of this octahedron on Σ_2 (with the "eraser-end of the pencil" at the origin) determines 8 subsurfaces of Σ_{d-1} having equal areas. At the next step we make the barycentric triangulation of each face and proceed to establish 48 subsurfaces having equal areas. In this way we produce a partition growing exponentially in cardinality, but compensated for by much finer estimates than (4.2). In particular, at the k^{th} step the partition $\{O_{j,k}\}$ has the properties that $j = 1,\cdots,2^3 6^{k-1}$ and $\mu_2(O_{j,k}) = \omega_2/(2^3 6^{k-1})$. Thus $\|\varphi_k\|_\infty \le (\omega_2\, 2^3 6^{k-1})^{1/2}$ and, hence, we have the estimate (for verifying (3.4)),

$$\frac{1}{n!}\sum_{k=2}^{n} \max(\|\varphi_{k-1}\|_\infty, \|\varphi_k\|_\infty)\, \max(\|\varphi_k\|_\infty, \|\varphi_{k+1}\|_\infty) \le \frac{\omega_2 2^3}{n!}\sum_{k=0}^{n} 6^k \le \frac{8\omega_2}{5n!}\, 6^{n+1},$$

which tends to 0. For this partitioning, our verification of convergence in the Levy topology also applies since the diameters of the $O_{j,k}$ tend to 0 as $k \longrightarrow \infty$.

5. Generalized harmonic analysis and spectrum estimation

Definition 5.1. Given $\varphi \in L^2_{loc}(\mathbb{R}^d)$ and define

$$\forall R > 0, \quad P_{\varphi,R} = \frac{1}{|B(R)|}\,(\varphi\chi_{B(R)}) * (\varphi\chi_{B(R)})^{\sim}$$

so that $P_{\varphi,R} \in L^1_{loc}(\mathbb{R}^d) \subseteq M(\mathbb{R}^d)$. Suppose that there is a continuous positive definite function P_φ for which $\lim_{R\to\infty} P_{\varphi,R} = P_\varphi$ in the $\sigma(M(\mathbb{R}^d), C_c(\mathbb{R}^d))$ topology. Then $P_\varphi \in L^\infty(\mathbb{R}^d)$ is the autocorrelation of φ and $\hat{P}_\varphi = \mu_\varphi$ is the power spectrum of φ.

Given the vague convergence, $\lim_{R\to\infty} P_{\varphi,R} = P_\varphi$, the positive definiteness of P_φ can be easily verified.

Suppose the data characterizing a given signal φ is known. In the following result, $\hat{f}$ can be thought of as a properly shaped window function so that the left side of (5.1) represents the power of φ in the region $\operatorname{supp}\hat{f}$. Formula (5.1) provides a method for computing this power in terms of the known functions f and φ. In practice, then, numerical estimates of the right hand side of (5.1) lead to a spectrum estimation algorithm.

Theorem 5.2. Given $\varphi \in L^2_{loc}(\mathbb{R}^d)$ with autocorrelation P_φ and power spectrum μ_φ. Assume there is an increasing function $i(R)$ on $(0,\infty)$ for which $\sup_{|x|\le R} |\varphi(x)| \le i(R)$ and $\lim_{R\to\infty} i(R)^2/R = 0$. Then

$$(5.1)\qquad \forall f \in C_c(\mathbb{R}^d),\ \int |\hat{f}(\gamma)|^2 d\mu_\varphi(\gamma) = \lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R)} |f*\varphi(t)|^2 dt.$$

(A simple estimate shows that $f*\varphi \in L^2_{loc}(\mathbb{R}^d)$ for $f \in C_c(\mathbb{R}^d)$ and $\varphi \in L^2_{loc}(\mathbb{R}^d)$.)

Proof. a. By Bochner's theorem we have $\mu_\varphi \in M_{b+}(\hat{\mathbb{R}}^d)$; and, by the $\sigma(M(\mathbb{R}^d), C_c(\mathbb{R}^d))$ convergence $\lim_{R\to\infty} P_{\varphi,R} = P_\varphi$ and the Parseval relation, we have

$$(5.2)\qquad \forall f \in C_c(\mathbb{R}^d),\ \lim_{R\to\infty} \int \frac{1}{|B(R)|} |(\varphi\chi_{B(R)})\hat{}(\gamma)|^2 |\hat{f}(\gamma)|^2 d\gamma = \int |\hat{f}(\gamma)|^2 d\mu_\varphi(\gamma).$$

Parseval can be applied since $\overline{(f*\tilde{f})}P_\varphi \in L^1(\mathbb{R}^d)$.

Using Plancherel's theorem we see that the left hand side of (5.2) is

$$(5.3)\qquad \lim_{R\to\infty} \frac{1}{|B(R)|} \int |f*(\varphi\chi_{B(R)})(t)|^2 dt;$$

in particular this limit exists for each $f \in C_c(\mathbb{R}^d)$. We complete the proof once we show that (5.3) equals

$$\lim_{R\to\infty} \frac{1}{|B(R)|} \int_{B(R)} |f*\varphi(t)|^2 dt; \tag{5.4}$$

in particular, we prove that this limits exists, noting that $f*\varphi \in L^2_{loc}(\mathbb{R}^d)$.

b. Using the triangle inequality for $L^2(\mathbb{R}^d)$ and the generalized Minkowski inequality, we have the estimate

$$\left| \left[\frac{1}{|B(R)|} \int |f*(\varphi\chi_{B(R)})(t)|^2 dt\right]^{1/2} - \left[\frac{1}{|B(R)|} \int |(f*\varphi)(t)\chi_{B(R)}(t)|^2 dt\right]^{1/2} \right|$$

$$\leq \frac{1}{|B(R)|^{1/2}} \left[\int |f*(\varphi\chi_{B(R)})(t) - (f*\varphi)(t)\chi_{B(R)}(t)|^2 dt\right]^{1/2}$$

$$= \frac{1}{|B(R)|^{1/2}} \left[\int \left| \int f(x)\left[(\varphi\chi_{B(R)})(t-x) - \chi_{B(R)}(t)\varphi(t-x)\right]dx \right|^2 dt\right]^{1/2} \tag{5.5}$$

$$\leq \frac{1}{|B(R)|^{1/2}} \int \left[\int \left| f(x)\left[(\varphi\chi_{B(R)})(t-x) - \chi_{B(R)}(t)\varphi(t-x)\right] \right|^2 dt\right]^{1/2} dx$$

$$= \frac{1}{|B(R)|^{1/2}} \int_{\text{supp } f} |f(x)| \left[\int \left| (\varphi\chi_{B(R)})(t-x) - \chi_{B(R)}(t)\varphi(t-x) \right|^2 dt\right]^{1/2} dx,$$

for each $f \in C_c(\mathbb{R}^d)$. Consequently, the proof of the result is complete once we show that the right hand side of (5.5) tends to 0 as $R \longrightarrow \infty$.

c. Let $\varphi(R,t-x) = (\varphi\chi_{B(R)})(t-x) - \chi_{B(R)}(t)\phi(t-x)$. Then $\varphi(R,t-x)$ is 0 if $t \in B(R)$ and $t \in x + B(R)$ or if $t \notin B(R)$ and $t - x \in B(R)$. Also, $\varphi(R,t-x) = \varphi(t-x)$ if $t \notin B(R)$ and $t \in x + B(R)$, and $\varphi(R,t-x) = -\phi(t-x)$ if $t \in B(R)$ and $t - x \notin B(R)$. Thus, the right hand side of (5.5) is

$$(5.6)\qquad \frac{1}{|B(R)|^{1/2}} \int_{\text{supp } f} |f(x)| \left(\int_{X_{x,R}} |\varphi(t-x)|^2 dt + \int_{Y_{x,R}} |\varphi(t-x)|^2 dt \right)^{1/2} dx,$$

where $X_{x,R} = \{t : t \notin B(R) \text{ and } t \in x + B(R)\}$ and $Y_{x,R} = \{t : t \in B(R) \text{ and } t \notin x + B(R)\}$.

If $R_f = \sup \{|x| : x \in \text{supp } f\}$, then straightforward calculations show that $X_{x,R} \subseteq B(R+R_f) \setminus B(R)$ and $Y_{x,R} \subseteq B(R) \setminus B(R-R_f)$ for $x \in \text{supp } f$ and large R. Consequently, we can make the estimate

$$\sup_{x \in \text{supp } f} \left(\int_{X_{x,R}} |\varphi(t-x)|^2 dt + \int_{Y_{x,R}} |\varphi(t-x)|^2 dt \right)^{1/2}$$

$$(5.7)\qquad \leq \sup_{x \in \text{supp } f} \Big[\left(|B(R+R_f)| - |B(R)| \right) \sup_{t \in B(R+R_f) \setminus B(R)} |\varphi(t-x)|^2$$

$$+ \left[|B(R)| - |B(R-R_f)| \right] \sup_{t \in B(R) \setminus B(R-R_f)} |\varphi(t-x)|^2 \Big]^{1/2}$$

$$\leq a(\varphi, R, f) \left(|B(R+R_f)| - |B(R-R_f)| \right)^{1/2},$$

where

$$a(\varphi, R, f) = \sup_{x \in \text{supp } f} \left[\sup_{t \in B(R+R_f) \setminus B(R)} |\varphi(t-x)|^2 + \sup_{t \in B(R) \setminus B(R-R_f)} |\varphi(t-x)|^2 \right]^{1/2}.$$

By the mean value theorem we have the estimate,

$$(5.8)\qquad \left[\frac{|B(R+R_f)| - |B(R-R_f)|}{|B(R)|} \right]^{1/2} \leq \left[\frac{2dR_f}{R} \right]^{1/2} \left[\frac{R+R_f}{R} \right]^{\frac{d-1}{2}}.$$

Combining (5.6), (5.7), and (5.8), we see that the right hand side of (5.5) is bounded by

$$|\text{supp } f|\ \|f\|_\infty \left(\frac{2dR_f}{R}\right)^{1/2}\left(\frac{R+R_f}{R}\right)^{\frac{d-1}{2}}\ a(\varphi,R,f);$$

and this will tend to 0, thus completing the proof, once we show that $a(\varphi,R,f)/R^{1/2}$ tends to 0 as $R \to \infty$.

If $t \in B(R+R_f)\backslash B(R)$ and $x \in \text{supp } f$ then $|t-x| \le R + 2R_f$, and if $t \in B(R)\backslash B(R-R_f)$ and $x \in \text{supp } f$ then $|t-x| \le R + R_f$. Therefore, $a(\varphi,R,f) \le \sqrt{2}\ i(R+2R_f)$, and so

$$\frac{a(\varphi,R,f)}{R^{1/2}} \le \frac{\sqrt{2}\ i(R+2R_f)}{(R+2R_f)^{1/2}}\left(\frac{R+2R_f}{R}\right)^{1/2}$$

which tends to 0 by hypothesis.

q.e.d.

Corollary 5.3. Given the hypotheses of Theorem 5.2. If $f \in L^p(\mathbb{R}^d)$ and $\varphi \in L^{p'}(\mathbb{R}^d)$, $1 \le p < \infty$, then the tempered distribution $\hat{f}$ is a well-defined element of $L^2_{\mu_\varphi}(\hat{\mathbb{R}}^d)$ and

$$\|\hat{f}\|_{2,\mu_\varphi} \le \|f\|_p \|\varphi\|_{p'}. \tag{5.9}$$

Proof. If $f \in C_c(\mathbb{R}^d)$ then (5.9) is a consequence of (5.1) and Holder's inequality. (5.9) extends to all of $L^p(\mathbb{R}^d)$ since $\overline{C_c(\mathbb{R}^d)} = L^p(\mathbb{R}^d)$, $1 \le p < \infty$; as such, (5.9) simultaneously defines $\hat{f}$ and gives a quantitative norm bound (of course, $\hat{f}$ is known to exist as a tempered distribution since the elements of $L^p(\mathbb{R}^d)$ are tempered).

q.e.d.

Remark 5.4. a. One can formulate a version of Theorem 5.2 for $L^{\infty}(\mathbb{R}^d)$ instead of $L^2_{loc}(\mathbb{R}^d)$. In this case the autocorrelation can be defined in terms of the weak * $\sigma(L^{\infty}(\mathbb{R}^d), L^1(\mathbb{R}^d))$ topology. As such, the analogue of Theorem 5.2 does not require any hypothesis involving $i(R)$; and the proof is much simpler since part c (of the proof) can be replaced by a simple estimate and application of Lebesgue dominated convergence, cf., [Me; B, 89-92].

b. For $\varphi \in L^2_{loc}(\mathbb{R}^d)$, the positive definite approximant $P_{\varphi,R}$ of Section 5 should be compared with the approximant

$$Q_{\varphi,R}(t) = \frac{1}{|B(R)|}\int_{B(R)} \varphi(t+x)\overline{\varphi(x)}dx$$

of Section 3. First, each $P_{\varphi,R}$ and $Q_{\varphi,R}$, $R > 0$, is continuous, and the bounds

$$|P_{\varphi,R}(t)| \leq \frac{1}{|B(R)|}\int_{B(R)} |\varphi(x)|^2 dx$$

and

$$|Q_{\varphi,R}(t)| \leq \frac{1}{|B(R)|}\left(\int_{B(R)} |\varphi(x)|^2 dx\right)^{1/2}\left(\int_{B(R)} |\tau_{-t}\varphi(x)|^2 dx\right)^{1/2}$$

are valid for all $t \in \mathbb{R}^d$. Second, if φ satisfies the growth condition (in terms of $i(R)$) given in Theorem 5.2 then $\{P_{\varphi,R} : R > 0\}$ and $\{Q_{\varphi,R} : R > 0\}$ have the expected similar behavior at infinity. More precisely, we can show that if $\lim_{R\to\infty} P_{\varphi,R} = P_{\varphi}$ (resp., $\lim_{R\to\infty} Q_{\varphi,R} = P_{\varphi}$) in the $\sigma(M(\mathbb{R}^d), C_c(\mathbb{R}^d))$ topology then $\lim_{R\to\infty} Q_{\varphi,R} = P_{\varphi}$ (resp., $\lim_{R\to\infty} P_{\varphi,R} = P_{\varphi}$) in the $\sigma(M(\mathbb{R}^d), C_c(\mathbb{R}^d))$ topology, and, in either case, P_{φ} is positive definite. There is a corresponding pointwise result.

6. An elementary restriction theorem

We begin with a straightforward Fourier transform weighted norm inequality having a measure weight.

Proposition 6.1. Given $\mu \in M_{b+}(\hat{\mathbb{R}}^d)$ for which $\mu^{\vee} = P \in L^{p'}(\mathbb{R}^d)$, $p \in [1,\infty]$. Then

$$\forall f \in L^1(\mathbb{R}^d) \cap L^p(\mathbb{R}^d), \quad \|\hat{f}\|_{2,\mu} \leq$$

(6.1)

$$\|P\|_{p'}^{1/2} (\|f\|_1 \|f\|_p)^{1/2} \leq (\tfrac{1}{2}\|P\|_{p'}^{1/2})(\|f\|_1 + \|f\|_p),$$

where $(L^1(\mathbb{R}^d) \cap L^p(\mathbb{R}^d), \|\cdots\|_1 + \|\cdots\|_p)$ is a Banach space.

Proof. The Parseval relation,

$$\forall f \in L^1(\mathbb{R}^d), \quad \int |\hat{f}(\gamma)|^2 \, d\mu(\gamma) = \int \overline{f * \tilde{f}(t)} P(t) dt, \tag{6.2}$$

is valid since $\overline{f * \tilde{f}(t)} P(t) \in L^1(\mathbb{R}^d)$. The right hand side of (6.2) is

$$\int \overline{f(x)} \left[\int f(x-t) P(t) dt \right] dx \leq \int |f(x)| \, \|f\|_p \, \|P\|_{p'} \, dx.$$

This and the arithmetic-geometric means inequality yield the result.

q.e.d.

The following is our elementary restriction theorem with explicit norm constant.

Proposition 6.2. Given $d \geq 2$ and $1 \leq p < 2d/(d+1)$. Then $\|\mu_{d-1}^{\vee}\|_{p'} < \infty$ and, for each $f \in L^1(\mathbb{R}^d) \cap L^p(\mathbb{R}^d)$,

$$(6.3)\qquad \left(\int_{\Sigma_{d-1}} |\hat{f}(\theta)|^2 \, d\sigma_{d-1}(\theta)\right)^{1/2} \le \left(\tfrac{1}{2}\|\check{\mu}_{d-1}\|_{p'}^{1/2}\right)\left(\|f\|_1 + \|f\|_p\right).$$

Proof. Using (4.1) we first show that $\check{\mu}_{d-1} \in L^{p'}(\mathbb{R}^d)$ for $1 \le p < 2d/(d+1)$. Not only does $J_\nu(s) = O(s^{-1/2})$, $s \to \infty$, e.g., Lemma 3.1, but, also, $J_\nu(s) \sim s^\nu/[2^\nu\Gamma(\nu+1)]$, $s \to 0$, $\nu > -1$. Thus, we can estimate

$$\|\check{\mu}_{d-1}\|_{p'}^{p'} = \omega_{d-1}(2\pi)^{p'}\left[\int_0^1 \cdots + \int_1^\infty r^{d-1}\, r^{-(\frac{d-2}{2})p'}\left|J_{\frac{d-2}{2}}(2\pi r)\right|^{p'} dr\right]$$

as follows. The first integral is finite (for any $p > 1$) by the stated asymptotic property of J_ν. The second integral is finite if $d - (dp'/2) + (p'/2) < 0$ and this follows if $1 < p < 2d/(d+1)$ and $d \ge 2$. The case $p = 1$ must be treated separately but is trivial.

The result follows from Proposition 6.1.

q.e.d.

Remark 6.3. a. In light of Theorem 4.1, the surface measure μ_{d-1} is the power spectrum of a signal φ, e.g., Definition 5.1 and Remark 5.4b. Since the norm constant in Proposition 6.2 is explicit and computable by (4.1), we see that (6.3) provides a means of estimating an upper bound for the power of φ in the region $\operatorname{supp}\hat{f}$, even though we don't have precise knowledge of φ itself.

b. In Proposition 6.1 where $\mu \in M_{b+}(\hat{\mathbb{R}}^d)$ is given, if $p = 1$ then P is always an element of $L^\infty(\mathbb{R}^d)$, and if $p = 2$ then $P \in L^2(\mathbb{R}^d)$ so that $\mu \in L^1(\hat{\mathbb{R}}^d) \cap L^2(\hat{\mathbb{R}}^d)$. In the case $\mu = \mu_{d-1}$ it is clear that $p = \infty$ can not be used in Proposition 6.1 since $\check{\mu}_{d-1} \notin L^1(\mathbb{R}^d)$.

Bibliography

[Ba] J. Bass, Fonctions de corrélation fonctions pseudo-aléatoires et applications, Masson, Paris, 1984.

[B] J. Benedetto, Spectral synthesis, Academic Press, NY, 1975.

[BH] J. Benedetto and H. Heinig, "Fourier transform inequalities with measure weights," Advances in Math. (to appear).

[Be] J.-P. Bertrandias, "Espaces de fonctions continues et bornées en moyenne asymptotique d'ordre p," Memoire Soc. Math. France 5(1966).

[Bo] N. Bourbaki, Intégration Livre VI, Hermann, Paris, 1952.

[CD] Y.-C. Chang and K. Davis, Lectures on Bochner-Riesz means, London Math. Soc. Lecture Notes, Series 114, Cambridge University Press, 1987.

[Ma] P. Malliavin, Intégration et probabilités, analyse de Fourier et analyse spectrale, Masson, Paris, 1982.

[Me] Y. Meyer, "Le spectre de Wiener," Studia Math. 27(1966), 189-201.

[S] E. Stein, "Oscillatory integrals in Fourier analysis," Beijing Lectures in Harmonic Analysis, Annals. of Math. Studies, Princeton University Press, 1986.

[T] P. Tomas, "Restriction theorems for the Fourier transform in harmonic analysis in Euclidean spaces," Proc. Symp. in Pure Math. 35, part 1 (1979), 111-114.

[W] N. Wiener, Collected works, volume II, P. Masani, ed., The MIT Press, 1979.

[WW] N. Wiener and A. Wintner, "On singular distributions," J. Math. and Phys. 17(1939), 233-246. (Collected works, volume II, P. Masani, ed.)

Prometheus Inc.
Final Report
11 July 1990

Section IV
Antenna-Polynomial Problems

Problems on Polynomials with Restricted Coefficients Arising from Questions in Antenna Array Theory

J.S. Byrnes

We present mathematical formulations of some classical problems in antenna design. For references concerning these problems, see the paper of Newman and Giroux in this volume.

To begin, we give precise mathematical interpretations to two standard electrical engineering definitions, that of *maximum sidelobe level* and *beamwidth*, which occur in the analysis and synthesis of a line array of equally-spaced, identical, omnidirectional antenna elements. As is well known, the model for such an array is a polynomial, where its degree is one less than the number of elements, and the coefficients are the *weights*, or *shading coefficients*, of the array.

Throughout this discussion, let $z = e^{i\theta}$ lie on the unit circle Γ, let $P(z)$ denote a polynomial with coefficients a_j, at least two of which are nonzero, and let θ_0 be the smallest nonnegative value of θ where the maximum of $|P(z)|$ occurs. Now let

$$\delta_1 = \min\{\, \delta : \delta < \theta_0 \text{ and } |P(e^{i\theta})| \text{ is increasing on } (\delta, \theta_0) \,\},$$

$$\delta_2 = \max\{\, \delta : \delta > \theta_0 \text{ and } |P(e^{i\theta})| \text{ is decreasing on } (\theta_0, \delta) \,\},$$

$$\sigma_P = \max_{\theta \in [\theta_0 - \pi, \theta_0 + \pi]/(\delta_1, \delta_2)} |P(e^{i\theta})|, \qquad \text{and}$$

$$\beta_P = \delta_2 - \delta_1.$$

The graph of (one full period of) $|P(e^{i\theta})|$ is the (modified) *beam pattern*, σ_P is the (modified) *maximum sidelobe level*, and β_P is the (modified) *beamwidth* of the array. We use the adjective "modified" because we have omitted the change of variable, normalization, and expression in terms of decibels that one finds in the engineering literature, but these differences are of no import here.

All problems presented will be of the following form:

Determine the existence and construction of a P which lies in a certain class, whose beam pattern satisfies certain properties, and which is optimum (or nearly so) with respect to one of its parameters. These classes are:

$$\mathcal{U} = \{\, P : |a_j| = 1 \,\} \quad \text{(the } \textit{unimodular} \text{ polynomials)},$$

$$\mathcal{U}_0 = \{\, P : |a_j| = 1 \text{ or } 0 \,\},$$

$$\mathcal{E} = \{\, P : a_j = \pm 1 \,\},$$

$$\mathcal{E}_0 = \{\, P : a_j = \pm 1, 0 \,\},$$

$$\mathcal{D}_K = \{\, P : \max |\frac{a_j}{a_m}| \le K \,\} \quad \text{(this maximum is the } \textit{dynamic range} \text{ of the array)},$$

$$\mathcal{D}_{K,0} = \{\, P : \max_{a_j \times a_m \neq 0} |\frac{a_j}{a_m}| \le K \,\},$$

$$\mathcal{C} = \text{any one of the above classes.}$$

Although a reasonable bound on the dynamic range is ordinarily important (an old rule of thumb is that "$K = 2$ and everyone is happy, $K = 10$ and some are happy, $K = 100$ and nobody is happy"), which would seem to render zero coefficients fatal, this is not usually the case. Since a zero coefficient simply means that that element is turned off (either by

choice or because it is not functioning properly), this does not affect the dynamic range in a meaningful way. Furthermore, it appears that in many cases allowing zero coefficients enables a significant reduction in the degree of P (see problem 4 below).

The time has come to state specific problems:

1. Given a finite (possibly empty) subset S of Γ, $q = e^{i\theta_0} \notin S$, $\sigma > 0$, $\beta > 0$, and a class $\mathcal{C}$, find $P \in \mathcal{C}$ of minimum degree (if it exists at all) such that $P(z) = 0$ for all $z \in S$, $\sigma_P \leq \sigma$, and $\beta_P \leq \beta$.

2. Given $\epsilon > 0$ and a class $\mathcal{C}$, find $P \in \mathcal{C}$ of minimum degree such that

$$\big||P(z_1)| - |P(z_2)|\big| \leq \epsilon \quad \text{for all } z_1, z_2 \in \Gamma$$

(a so-called *ultra-flat* polynomial).

3. Given $S = \{e^{i\theta_1}, e^{i\theta_2}, \ldots, e^{i\theta_m}\}$ nonempty, $\epsilon > 0$, $\delta > 0$, and a class $\mathcal{C}$, find $P \in \mathcal{C}$ of minimum degree such that $P(z) = 0$ for all $z \in S$ and

$$\big||P(z_1)| - |P(z_2)|\big| \leq \epsilon \quad \text{for all } z_1 = e^{i\tau_1}, z_2 = e^{i\tau_2}$$

with the property that $(\tau_1, \tau_2) \cap (\theta_j - \delta, \theta_j + \delta)$ is empty for $1 \leq j \leq m$ (a *notch filter*).

It is appropriate to observe that, although optimization of the degree is called for in problems 1–3, near-optimal solutions are definitely of interest. For example, the problem of most practical interest is 1, in the case where the cardinality of S is very small (even 0, 1, or 2), and this appears mathematically interesting as well. Also along these lines, we have:

4. Prove or disprove the conjecture: The $P \in \mathcal{U}$ of minimum degree with an n-fold zero at $z = 1$ is

$$\prod_{m=0}^{n-1} (1 - z^{2^m}).$$

In fact, even a proof that the degree of such a P must be exponential in n would be of interest. Note that if the class $\mathcal{U}$ is replaced by $\mathcal{U}_0$ (actually even $\mathcal{E}_0$), then there is a P of degree less than n^3 satisfying the required property.

Analogous problems, when one of the other parameters aside from the degree is optimized, are of interest as well. In this case there is an additional class of polynomials to be considered:

$$\mathcal{M}_n = \{\, P : \text{degree } P = n \text{ and } |a_j|, 0 \leq j \leq n, \text{ are preassigned} \,\}.$$

Thus, another typical problem is:

5. Given $q = e^{i\theta_0}$, $\beta > 0$, the degree n (often between 10 and 40, although values can be as much as a few thousand), and the class $\mathcal{C}$ of P, find P such that $\beta_P \leq \beta$ and σ_P is minimized.

The simple job of stating other varieties of these problems is left to the reader. Their solution, a much more daunting task, also awaits the reader's efforts.

Prometheus Inc.
Final Report
11 July 1990

Section V
An Ideal Peak Factor Array

An ideal omnidirectional transmitting array, and optimal peak factor array, for less than half-wavelength spacing

James S. Byrnes †
Prometheus Inc.
21 Arnold Avenue
Newport, RI 02840

Abstract

We give an explicit construction of an ideal omnidirectional transmitting array, and optimal peak factor array, for the case of a linear array of identical omnidirectional elements with uniform spacing of less than half-wavelength. The construction is based upon the Byrnes polynomials, introduced by the author in 1977.

Introduction

A consideration which is often important in the synthesis of an antenna pattern is the *peak factor*, which is the ratio of the peak to average power of the array. For a line array of equally spaced omnidirectional elements, the mathematical model for the pattern is a polynomial whose coefficients are the *weights*, or *shading coefficients*, of the array. The quantity to be synthesized in this case is the magnitude of $P(z)$ on the unit circle $\mathcal{C} = \{z : |z| = 1\}$, and the peak factor is now the ratio of the *sup* norm to the L^2 norm of P.

Clearly the peak factor is bounded below by 1, and the classical problem is to make it as close to 1 as possible. In addition to arising in peak power limited transmitting and other aspects of antenna design, the identical question occurs in digital filtering. Furthermore, the same polynomial problem has been the object of intense study by mathematical analysts for more than fifty years, including such notable mathematicians as P. Erdös [3], G.H. Hardy, J.P. Kahane [4], T. Körner [5], J.E. Littlewood [6], and D.J. Newman [7].

A parallel concern, to engineers and mathematicians alike, is the question of synthesizing various antenna patterns (i.e., constructing polynomials with specified moduli on the unit circle) when certain restrictions are placed upon the coefficients. When transmitting, for example, one usually wants to maximize the total power output of the array, which is achieved when each individual element broadcasts at full power. Thus, in this case, the coefficients must have the same magnitude. In addition, the usual peak power limitation means that the output power must be as close to contact as possible in all directions, i.e., $|P(z)|$ should be as close to constant as possible on $\mathcal{C}$. The fundamental result in this specific area is that of Kahane [4]:

Theorem. There is an absolute constant $c > 0$ such that for each positive integer n, there is a polynomial $P(z)$ of degree n with unimodular coefficients which satisfies

$$|P(z)| = \sqrt{n} + E \quad \text{for all } z \in \mathcal{C}, \quad \text{where } |E| \leq cn^{\frac{1}{10}} \log n. \tag{1}$$

Since any such P has $n+1$ coefficients of modulus 1, so that its L^2 norm is obviously $\sqrt{n+1}$, this shows that the ideal peak factor of 1 is indeed achievable asymptotically, even when the coefficients are required to have the same magnitude. Note that Kahane's theorem shows that the ratio of the maximum modulus of P to the minimum modulus of P can be asymptotically 1 as well, a stronger result than the peak factor one just mentioned. As an aside, we mention that the corresponding problem when the coefficients are required to be ± 1 is one of the important unsolved problems in this area of mathematical analysis.

Although Kahane's beautiful result was greeted with much enthusiasm by the mathematical community, it has not found application in the engineering problems discussed above. This is because his proof is an *existence* one: he shows probabilistically that such polynomials must occur. Thus far such functions have not been actually constructed. This too is an important outstanding question, in both mathematics

† The author is also Professor of Mathematics at the University of Massachusetts at Boston

Research sponsored by the Air Force Office of Scientific Research (AFSC), under Contract F49620-88-C-0028.

and engineering. The purpose of this note is to give an *explicit* construction of an ideal omnidirectional transmitting array, and optimal peak factor array, for the case of a linear array of identical omnidirectional elements with uniform spacing of less than half-wavelength. In certain circumstances such close spacing can cause a notable increase in mutual coupling. This physical problem will not be addressed here.

Results

As is well known, for a linear array of n equally spaced identical omnidirectional elements the array factor is the polynomial

$$P(z) = \sum_{j=0}^{n-1} a_j z^j, \quad z = e^{2\pi i u}, \quad u = \frac{d \sin\theta}{\lambda} \quad . \tag{2}$$

Here, the a_j are the shading coefficients, d is the array spacing, λ is the wavelength, and θ is the angle of incidence ($\theta = 0$ is broadside).

Note that for $d = \lambda/2$, generally considered the ideal case, $2\pi u$ goes from $-\pi$ to π as $\sin\theta$ goes from -1 to 1, so that z traverses C completely. However, if $d < \lambda/2$, then a portion of C will be omitted by z. This is precisely the property that we now exploit.

In fact, the construction follows immediately from earlier work of the author [2]. The crucial property of the polynomials introduced there, and later called *Byrnes polynomials* [1, 5], is given by part (*ii*) of the basic theorem in [2]:

Theorem. For n a positive integer, let

$$P(z) = \sum_{k=0}^{n-1} \sum_{j=0}^{n-1} \exp(2\pi i j k n^{-1}) z^{j+kn}, \quad z = e^{2\pi i u}. \tag{3}$$

Then for any ε, $\quad n^{-1} < \varepsilon < \frac{1}{2}$, $\quad$ we have $\quad |P(z)| = n + E \quad$ for all u, $\quad \varepsilon < |u| \leq \frac{1}{2}$,
where $|E| < 1 + 2\pi^{-1} + 5(\pi\varepsilon)^{-1}$.

To clarify the significance of this result in the present context, suppose for simplicity that $d = \lambda/4$, so that z is now traversing exactly 1/2 of C as $\sin\theta$ goes from -1 to 1. The theorem immediately yields the correct choice of coefficients, gotten by merely changing z to $-z$ in (3). Hence, the optimal polynomial is

$$P(z) = \sum_{k=0}^{n-1} \sum_{j=0}^{n-1} (-1)^{j+kn} \exp(2\pi i j k n^{-1}) z^{j+kn} \quad z = e^{2\pi i u}, \quad u = \frac{\sin\theta}{4} \quad . \tag{4}$$

This gives $|P(z)| = n + E$, $|E| < 1 + 22\pi^{-1} < 9$ for **all** θ. Note that the degree of P is $n^2 - 1$, so that its L^2 norm is n and it represents an array with n^2 elements. Furthermore, the error E is *uniformly* bounded independent of n, so that we indeed have an optimal peak factor array. In addition, the coefficients of P are unimodular, so that it also represents an ideal omnidirectional transmitting array. These properties are illustrated in Figures 1.a-1.e, showing polar plots of $|P(z)|$ in Equation (4) for various choices of n and for $-\pi \leq \theta \leq \pi$. Observe that for small n the actual array patterns exhibit an error that is considerably smaller than that obtained for the general case considered in [2]. Finally, the spread (i.e., the difference between the maximum and minimum, expressed in dB's) of $|P(z)|$ as a function of the number n^2 of elements is shown in Figure 2. This graph clearly illustrates the desired flatness of $|P(z)|$, as described above.

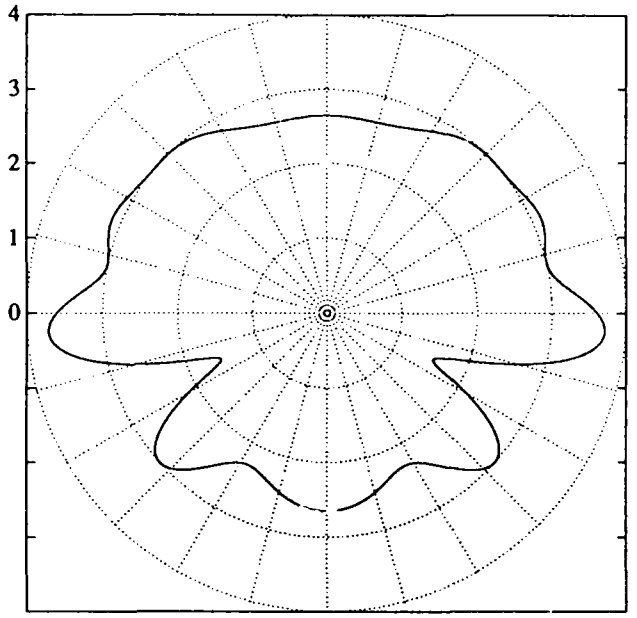


Figure 1.a: 9 elements (n=3)

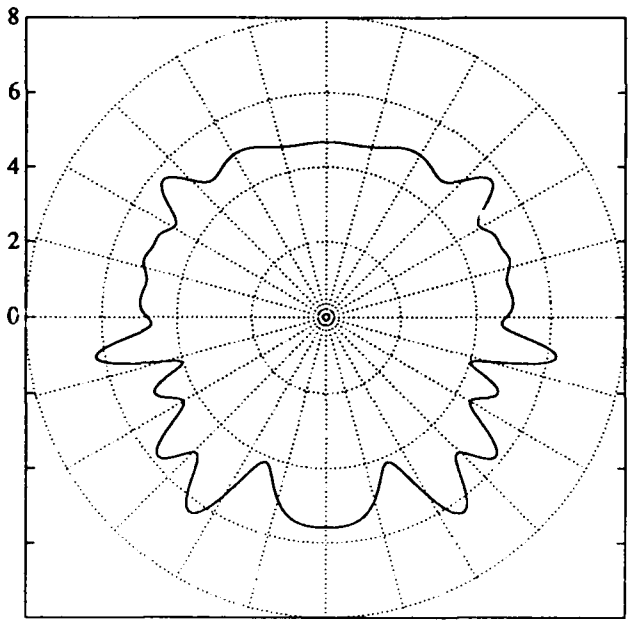


Figure 1.b: 25 elements (n=5)

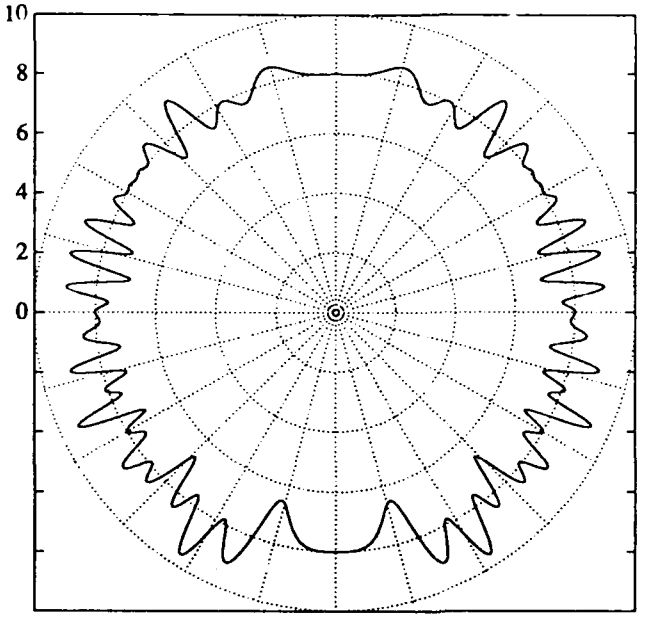


Figure 1.c: 64 elements (n=8)

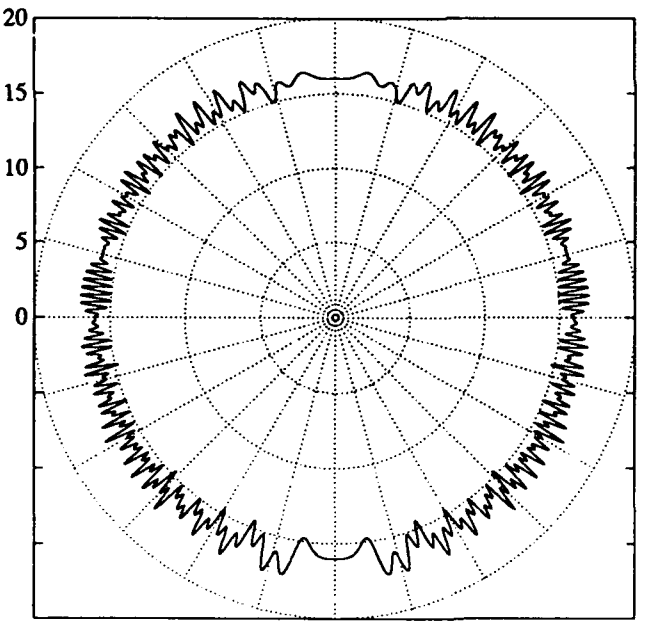


Figure 1.d: 256 elements (n=16)

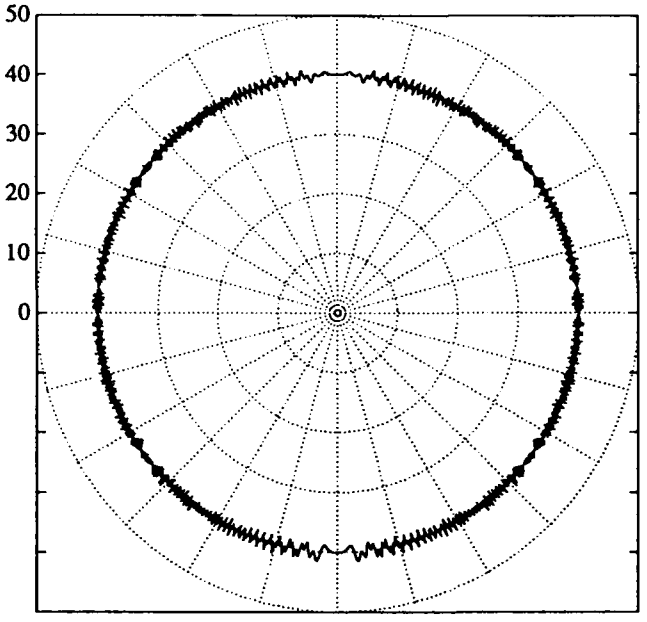


Figure 1.e: 1600 elements (n=40)

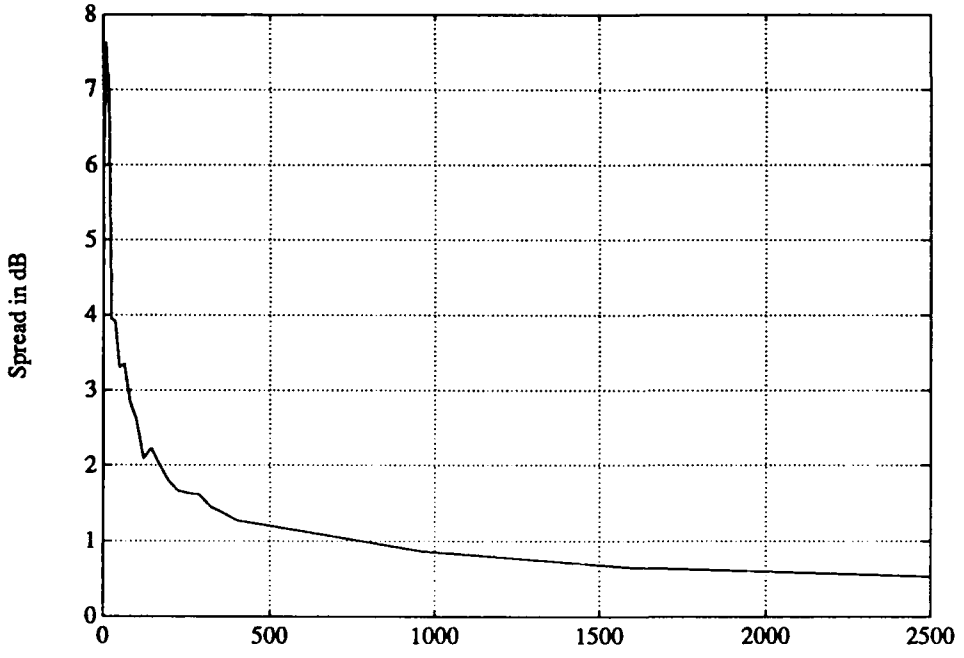


Figure 2: Spread of |P(z)| as a function of the number of elements.

Bibliography

[1] G. Benke, "On the maximum modulus for a certain class of unimodular trigonometric polynomials," in *Recent advances in Fourier analysis and its applications*, pp. 73-90, J. S. Byrnes and J. L. Byrnes, eds., Kluwer, 1990.

[2] J. S. Byrnes, "On polynomials with coefficients of modulus one," *Bull. London Math. Soc.*, vol. 9, pp. 171-176, 1977.

[3] P. Erdös, "Some unsolved problems," *Mich. Math. J.*, vol. 4, pp. 291-300, 1957.

[4] J.-P. Kahane, "Sur les polynômes à coefficients unimodulaires," *Bull. London Math. Soc.*, vol. 12, pp. 321-342, 1980.

[5] T. W. Körner, "On a polynomial of Byrnes," *Bull. London Math. Soc.*, vol. 12, pp. 219-224, 1980.

[6] J. E. Littlewood, "On the mean values of certain trigonometric polynomials," *J. London Math. Soc.*, vol. 36, pp. 307-334, 1961.

[7] D. J. Newman, "An L^1 extremal problem for polynomials," *Proc. Amer. Math. Soc.*, vol. 16, pp. 1287-1290, 1965.

Prometheus Inc.
Final Report
11 July 1990

Section VI
Polynomials with Unimodular Coefficients

Properties on the Unit Circle of Polynomials with Unimodular Coefficients

By Donald J. Newman and André Giroux†

Prometheus Inc.
21 Arnold Ave.
Newport, RI 02840

The following problem is posed in [1]: given the magnitude of the coefficients of a polynomial P, a finite subset S of the unit circle C, and a point p on C distinct from those in S, choose the phases of the coefficients so that $P(z) = 0$ for all z in S, the maximum on C of $|P(z)|$ occurs at $z = p$, and the maximum of $|P(z)|$ on a subset of C excluding an appropriate interval around p (the "beamwidth") is as small as possible. As explained in [1], this problem arises naturally in linear antenna theory, filter theory, and other classical electrical engineering applications.

Our main result is encompassed in the following theorem:

Theorem. Given $\alpha_1, \alpha_2, \ldots, \alpha_n, \beta$ on C, $\beta \neq \alpha_i$, there exists a polynomial P with coefficients of modulus one such that the zeros of P on C are precisely the points $\alpha_1, \alpha_2, \ldots, \alpha_n$ and that

$$\|P\| \equiv \max\{|P(z)| : z \in C\} = |P(\beta)|.$$

(The points α_j are not assumed to be distinct: multiple zeros are allowed.)

We shall prove the theorem in two steps, first constructing a polynomial with coefficients of modulus one with the prescribed zeros (lemma 1) and then modifying it so that it also has the prescribed maximum (lemma 4).

Let U denote the (non-linear) class of polynomials with coefficients of modulus one. We introduce in U the operation of "encapsulation" which we denote by $\otimes$.

Definition. Let M be the degree of P. Then

$$(P \otimes Q)(z) \equiv P(z)Q(z^{M+1}).$$

Explicitly, then, if $P(z) = \sum_{k=0}^{M} a_k z^k$ and $Q(z) = \sum_{j=0}^{N} b_j z^j$, then $P \otimes Q(z) = \sum_{i=0}^{N(M+1)+M} c_i z^i$, where $c_{j(M+1)+k} = a_k b_j$ for $0 \leq k \leq M$ and $0 \leq j \leq N$.

From this expression, it is clear that $\otimes$ is associative and that U is indeed closed under it. Also, if Q does not vanish on C, then the zeros of $P \otimes Q$ are precisely those of P.

Lemma 1 . Given $\alpha_1, \alpha_2, \ldots, \alpha_n$ on C, there exists P_n in U which vanishes at the α_j and nowhere else on C.

Proof. Let $\omega = \exp 2\pi i/3$. We shall prove a little more, namely that such a polynomial can be found with constant term one and leading term $-\omega$. We use induction on n. Let

$$Q_1(z) = 1 - z/\alpha_1$$

and let

$$P_1(z) = Q_1(z) \quad \text{if} \quad \alpha_1\omega = 1$$

and

$$\begin{aligned} P_1(z) &= Q_1(z) \otimes (1 + z + \alpha_1\omega z^2) \\ &= 1 - (1/\alpha_1)z + z^2 - (1/\alpha_1)z^3 + \alpha_1\omega z^4 - \omega z^5 \end{aligned}$$

† The authors are also with Temple University and the University of Montreal respectively.

otherwise (the condition $\alpha_1\omega \neq 1$ also ensures that the polynomial $1 + z + \alpha_1\omega z^2 \neq 0$ on C). Assume now that $R(z) = 1 + \cdots - \omega z^N$ is a polynomial in U of which the zeros on C are precisely the points $\beta_j = \alpha_j/\alpha_n$ for $1 \leq j \leq n-1$. Set

$$\begin{aligned} S(z) &= R(z)(1 - z^N + z^{2N} - z^{3N+1}) \\ &= 1 + \cdots + \omega z^{4N+1}. \end{aligned}$$

To verify that S is in U, we need only check the modulus of the coefficients of z^N and z^{2N}. But, the coefficient of z^N is $-\omega - 1 = \omega^2$ and that of z^{2N} is $\omega + 1 = -\omega^2$. We now claim that

$$1 - z^N + z^{2N} - z^{3N+1}$$

does not vanish on C except for a simple zero at $z = 1$. Indeed, for the sum of four points on C to vanish, it is necessary and sufficient that they cancel in pairs. There are thus three possibilities:

a) $1 - z^N = 0$ and $z^{2N} - z^{3N+1} = 0$; this gives $z^N = 1$ and $z^{N+1} = 1$, hence $z = 1$;
b) $1 + z^{2N} = 0, -z^N - z^{3N+1} = 0$; this implies $z^{2N} = -1, z^{2N+1} = -1$, which is impossible;
c) $1 - z^{3N+1} = 0, -z^N + z^{2N} = 0$; in this case, $z^{3N+1} = 1$ and $z^N = 1$, therefore $z^{2N+1} = z^{N+1} = z^N = 1$, hence $z = 1$ again.

Setting

$$Q_n(z) = S(z/\alpha_n) = 1 + \cdots + (\omega/\alpha_n^{4N+1})z^{4N+1}$$

we obtain a polynomial in U vanishing on C precisely at the points $\alpha_1, \alpha_2, \cdots, \alpha_n$. Then

$$P_n(z) = Q_n(z) \quad \text{if} \quad \alpha_n^{4N+1} = -1$$

and

$$P_n(z) = Q_n(z) \otimes (1 + z - \alpha_n^{4N+1} z^2) \quad \text{otherwise.}$$

This completes the induction and the proof of the lemma.

Lemma 2. If $n > 3$, the polynomial $p_n(z) = 1 + z + z^2 + \cdots + z^{n-1} - z^n + z^{n+1} + \cdots + z^{2n-1}$ does not vanish on C and $\max\{|p_n(z)| : z \in C\}$ is attained if and only if $z = 1$.

Proof. Since

$$p_n(z) = \frac{z^{2n} - 1}{z - 1} - 2z^n$$

we see that

$$p_n(e^{i2\theta})e^{-i(2n-1)\theta} = \frac{\sin 2n\theta}{\sin \theta} - 2e^{i\theta}$$

so that $p_n(e^{i2\theta}) = 0$ implies that $e^{i\theta}$ is real which implies in turn that $p_n(e^{i2\theta}) = p_n(1) = 2n - 2$, a contradiction.

Our statement about the maximum modulus property of $p_n(z)$ follows from the fact that ${p_n}^2(z)$ has positive coefficients.

Lemma 3. If $n > 0$, the polynomial $q_n(z) = 1 + z + z^2 + \cdots + z^n - z^{n+1}$ does not vanish on C.

Proof. One has $(1 - z)q_n(z) = 1 - 2z^{n+1} + z^{n+2}$, which, by the triangle inequality, can vanish only if $1, z^{n+1}$ and z^{n+2} are in the same direction; that is, here, $z = 1$. But, obviously, $q_n(1) \neq 0$.

Lemma 4. If P is any polynomial such that $P(\beta) \neq 0$, then there exists a polynomial T in U which does not vanish on C and is such that $P \otimes T$ attains its maximum modulus on C at $z = \beta$.

Proof. We can assume that $\beta = 1$. let $T = q_m \otimes r \otimes p_n$ where q_m and p_n are as in lemmas 3 and 2 respectively, and $r(z) = 1 + e^{i\theta}z - z^2$ with $\theta \neq \pm\pi/2$: $r(z)$ does not vanish on C and $\operatorname{Im} r'(1)/r(1) = 2\sin\theta$. Consider first $P^* = P \otimes q_m \otimes r$. If M is the degree of P, then

$$\frac{P^{*\prime}(1)}{P^*(1)} = \frac{P'(1)}{P(1)} + (M+1)\frac{q_m'(1)}{q_m(1)} + (M+1)(m+2)\frac{r'(1)}{r(1)}$$

and therefore

$$\operatorname{Im}\frac{P^{*\prime}(1)}{P^*(1)} = \operatorname{Im}\frac{P'(1)}{P(1)} + 0 + (M+1)(m+2)\operatorname{Im}\frac{r'(1)}{r(1)} = \operatorname{Im}\frac{P'(1)}{P(1)} + (M+1)(m+2)2\sin\theta.$$

A proper choice of m and θ will cancel this expression, making zero a stationary point of the function $|P^*(e^{i\theta})|^2$. Since this function does not vanish at $\theta = 0$, this stationary point can be transformed into the maximum point over all θ by multiplying P^* by a delta-like function, in our case by encapsulating it with p_n for suitably large n. Indeed, one has

$$|p_n(e^{i\theta})|^2 = 4 - 4\frac{\sin n\theta}{\sin\theta/2}\cos\theta/2 + \left(\frac{\sin n\theta}{\sin\theta/2}\right)^2,$$

By computing the second derivative, it is easily seen that $p_n(\theta)$ falls off like $e^{-cn\theta^2}$, and this shows that it is an adequate delta function for our purposes.

This completes the proof.

Reference.

[1] James S. Byrnes, Donald J.Newman, "Null Steering Employing Polynomials with Restricted Coefficents", *IEEE Transactions on Antennas and Propagation*, 36(1988), 301–303.

Prometheus Inc.
Final Report
11 July 1990

Section VII
awd *Overview*

COMPUTER-AIDED
ANTENNA ARRAY WEIGHT DESIGN

A SPECIFICATION LANGUAGE APPROACH

prepared by

Nasser Khraishi, Stephen Boyd, Richard Roy
Prometheus, Inc.
103 Mansfield St.
Sharon, MA 02067

also with
Information Systems Laboratory
Stanford University
Stanford, CA 94305

ANTENNA ARRAY WEIGHT DESIGN
OVERVIEW OF PRESENTATION

OVERVIEW OF PRESENTATION

- Sensor array signal processing — problem formulation
- Various objectives and approaches — real-time versus off-line
- Problem complexity and *realistic* simplifying assumptions
- Antenna array weight design
- `AWD` – the program
- Examples

ANTENNA ARRAY WEIGHT DESIGN
SENSOR ARRAY PROCESSING

General Problem Characteristics

- Inputs are discrete samples of a continuous function of one or more independent variables
  - digital filtering of analog signals
  - phased array radars
- The function being sampled generally consists of a superposition of ***multiple signals of interest*** and noise
  - sinusoids in analog signals – time series analysis
  - plane waves impinging on an array – sensor array processing
- Signals may be ***functions*** of parameters of interest
  - frequencies of sinusoids
  - directions-of-arrival (DOAs) of plane waves
- Reconstruction of signals as functions of *time* may be of primary interest

SENSOR ARRAY PROCESSING OBJECTIVES AND APPROACHES

Objectives

- Obtain parameter estimates in *real-time*
  - track time-varying frequencies of sinusoids in noise
  - track moving targets – active/passive radar tracking
- Obtain *optimal* parameter estimates *off-line*
  - trajectory reconstruction – error analysis
- Reconstruct one or more signals individually in multiple signal environment in *real-time*
  - establish reliable communication links
  - passive listening in multiple source environments

Approaches

- Off-line
  - many approaches possible
  - choice depends on computational cost and available resources (*cf.* maximum-likelihood)
- Real-time
  - *adaptive* array processing
  - *pseudo-adaptive* array processing

ANTENNA ARRAY WEIGHT DESIGN PROBLEM FORMULATION

Problem Statement

- GIVEN
  - sensor array of known fixed geometry,
  - statistics (second-order) of noise,
  - parameters of signals of interest (*e.g.* DOA, center frequency)
  - parameters of unwanted signals (*e.g.* jammers)
- FIND
  - a linear combination (weight vector/linear functional) of sensor element outputs

that

- MINIMIZES/MAXIMIZES
- – a cost/merit function of the combined array output
- SUBJECT TO
  - various design specifications.

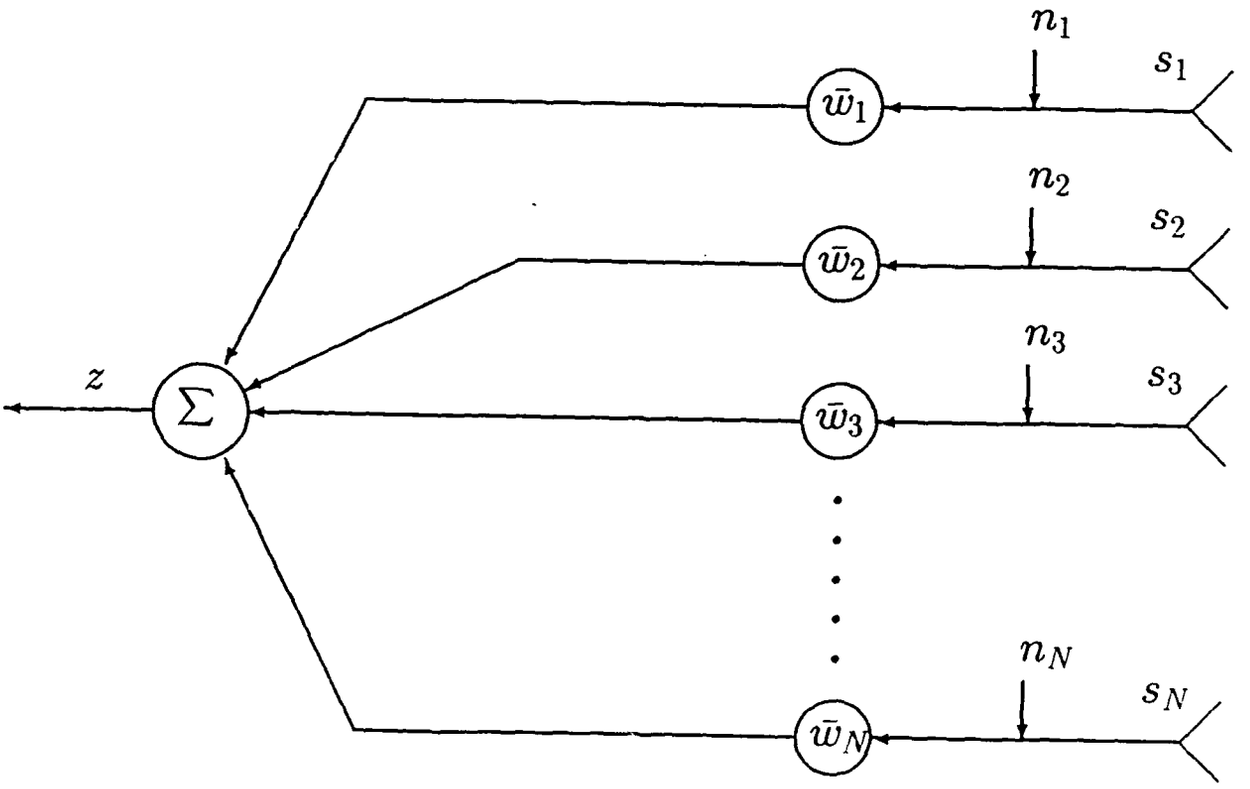
ANTENNA ARRAY WEIGHT DESIGN SIMPLIFYING ASSUMPTIONS

Simplifying Assumptions

- Medium of transmission is assumed to be isotropic and non–dispersive.
- Array consists of sensor elements with *fixed* relative locations.
- Signal sources are assumed to be farfield and lie in the plane of the array.
- Sensor elements are omni-directional.
- Elements have constant gain inside a predetermined (finite) frequency band of operation and negligible gain outside such band.
- Noise present in every sensor output is
  - additive, zero–mean, and stationary,
  - uncorrelated with the signals of interest (SOI).

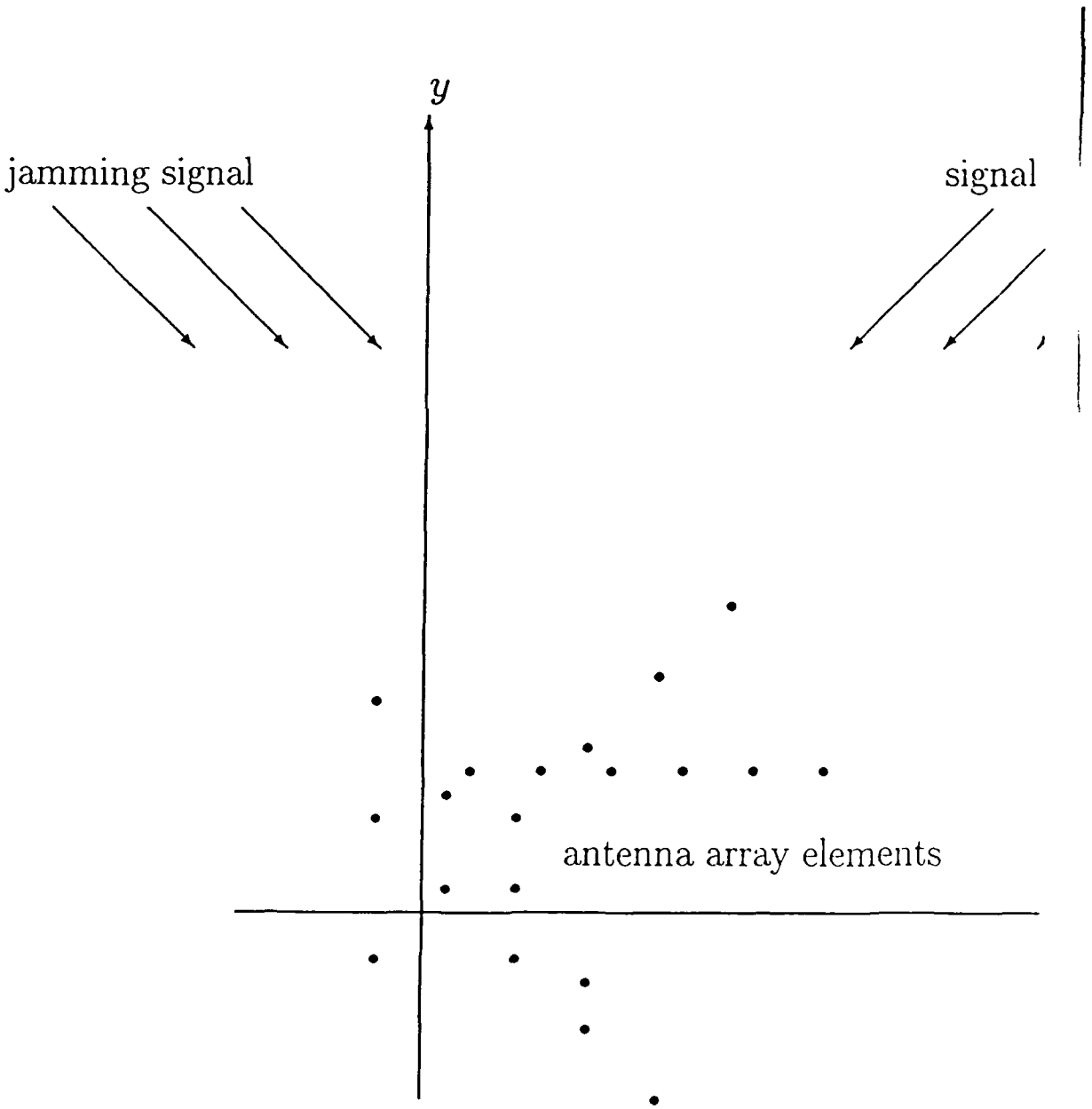
ANTENNA ARRAY WEIGHT DESIGN PROBLEM FORMULATION

Combined Array Output



ANTENNA ARRAY WEIGHT DESIGN PROBLEM FORMULATION

Array Under Consideration



ANTENNA ARRAY WEIGHT DESIGN PROBLEM FORMULATION

Basic Definitions

- Complex weight vector (to be designed)

$$w \equiv \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

- The noise–free sensor output vector

$$s \equiv \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix}$$

- The noise vector

$$n \equiv \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix}$$

ANTENNA ARRAY WEIGHT DESIGN PROBLEM FORMULATION

Basic Equations

- Noise–free output

$$S(\theta, \omega) = w^* s(\theta, \omega)$$

- Noise output

$$S_n = w^* n$$

- Total output

$$z(\theta, \omega) = S(\theta, \omega) + S_n$$

- Total noise power

$$P_n = \sum_{i=1}^{N} \sigma_i |w_i|^2 = w^* \Sigma w$$

where

$$\Sigma \equiv \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N) = E(nn^*)$$

ANTENNA ARRAY WEIGHT DESIGN PROBLEM FORMULATION

Sensor Element Output

- The i^{th} sensor element output can be written as

$$s_i(\theta, \omega) = \exp(j2\pi f(x_i \cos\theta + y_i \sin\theta))$$

- f is the normalized (dimensionless, $f_{\text{actual}}/f_{\text{nominal}}$) signal frequency,
- θ is the incident angle (degrees),
- x_i is the x-location (in units of λ_{nominal}) of element i, and
- y_i is the y-location (in units of λ_{nominal}) of element i.

ANTENNA ARRAY WEIGHT DESIGN
REAL–WORLD PROBLEMS

Real–World Problem Characteristics

- Signals of
  - unknown number,
  - unknown directions of arrival (DOAs),
  - unknown frequencies.
- Modeling errors
  - unmodeled signal environment (*e.g.* reflectors, echo, *etc.*),
  - inter–element effects,
  - anisotropic media,
  - nonlinearities.

ANTENNA ARRAY WEIGHT DESIGN HANDLING REAL–WORLD DESIGN

On–Line Adaptive Methods

- Maximizes (minimizes) a certain merit (penalty) function (*e.g.* SNR) of the combined array output.
- Rely on some knowledge of SOI,
  - DOA, frequency,
  - functional form.
- Difficult to take advantage of *apriori* knowledge of interfering signal frequencies, directions–of–arrival, *etc.*
- Designed to be computationally efficient and very fast.
- Basically places pattern nulls in interference DOA's. Thus, is quite sensitive to interference/signal correlation.

ANTENNA ARRAY WEIGHT DESIGN HANDLING REAL–WORLD DESIGN

Deterministic Methods

- Assumes knowledge of frequencies and directions of arrival of signals of interest and interference.
- Computationally demanding
  - slow compared to adaptive methods,
  - unsuitable for on–line compuation,
  - pre–storage of off–line results is necessary.
- Merit functions, pattern nulls, pattern constraints and array gains are arbitrary.
- Sensitivity to data can be reduced as desired by clever choice of design constraints and/or the merit function.

ANTENNA ARRAY WEIGHT DESIGN HANDLING REAL–WORLD DESIGN

Combining Adaptive/Deterministic Methods

- Adaptively estimate signal/interference parameters and obtain an initial set of weights.
- Select an appropriate merit (penalty) function and a desired set of desing constraints, and
  - run a suitable deterministic scheme (*e.g.* our scheme),
  - obtain ***optimal*** weight vector,
  - such weights produce desired output.
- One suggestion to use results of deterministic schemes on–line is to
  - devise a combination possible scenarios,
  - use off–line computations to find corresponding optimal weight vectors,
  - store resulting weight vectors in memory,
  - using on–line data, determine most appropriate weight vector to use.

ANTENNA ARRAY WEIGHT DESIGN
DETERMINISTIC ANTENNA WEIGHT DESIGN

Closed Form Methods

- Usually old methods (40's and 50's).
- Easy to implement (hand calculations, charts, ...)
- Restrictive in nature.
- Fast.
- Example: Dolph's method.

Iterative Methods

- Newer methods. Uses currently available computing power.
- Implementation requires intensive computations.
- Less restrictive.
- Much slower.
- Example: our method.

ANTENNA ARRAY WEIGHT DESIGN A HISTORY OF DETERMINISTIC METHODS

Closed Form Methods

- Stone (U. S. Patents 1,643,323 and 1,715,433):
  - A binomial expansion representaion.
  - Design of patterns by choice of coefficients.
- Schelkunoff (1943):
  - Polynomial representation of arrays of equally spaced antenna elements.
  - Design of patterns by cleverly placing zeroes of the representing polynomial.
- Dolph (1946):
  - Design using Tchebychev polynomials.
  - Obtained optimal tradeoff between sidelobe gain and mainlobe width.
  - First to talk about solutions that are *optimal* in some sense.

ANTENNA ARRAY WEIGHT DESIGN
A HISTORY OF ITERATIVE METHODS

Non–convex Design

- Cheng and Tseng (1965,1967):
  - Optimization of Hermitian ratios.
  - Unconstrained case: generalized eigenvalue problem.
  - Constrained case: iterative solutions.
  - Method actually known before. Others used same methods. For example, Butler, *et al.*, (1971,1972).
- Waren, *et al.*,(1966,1967):
  - General mathematical programming approach.
  - Optimization of magnitude and phase of weights.
  - Design of array geometry.

ANTENNA ARRAY WEIGHT DESIGN
A HISTORY OF ITERATIVE METHODS

Convex Design

- McMahon, *et al.*, (1972):
  - Optimize real and imaginary parts of weights taken separately.
  - Translate problem into linear program.
  - Fake a "tuned" objective funtion.
- Wilson (1976):
  - Similar to above approach.
  - Uses a Tchebychev–norm approximation technique.
- Evans and Fortmann (1975): a functional programming approach.

ANTENNA ARRAY WEIGHT DESIGN CONVEXITY

Basic Definitions

Definition 1 (Convex Set) *$\Omega \subseteq V$ is said to be a convex set if $\mu x + (1-\mu)y \in \Omega$ whenever $x, y \in \Omega$ and $0 \leq \mu \leq 1$.*

Definition 2 (Convex Function) *A function $f : V \rightarrow \Re$ is a convex function on a convex set $\Omega \subseteq V$ if $f(\mu x + (1-\mu)y) \leq \mu f(x) + (1-\mu)f(y)$ for all $x, y \in \Omega$ and $0 \leq \mu \leq 1$.*

Definition 3 (Convex Program)

$$\begin{array}{ll} \text{minimize} & \Phi(\omega) \\ \text{subject to} & \omega \in \Omega \end{array}$$

where $\Phi : V \rightarrow \Re$ is a convex function and Ω is a convex set.

ANTENNA ARRAY WEIGHT DESIGN CONVEXITY

Basic Facts

Fact 1 *Given the convex program described above, and given that the set Ω is non-empty and closed, then the set of points, $\Omega_0 \subseteq \Omega$, that solves the convex program is a non-empty convex set.*

Fact 2 *Let $f_1 : V \rightarrow \Re$ and $f_2 : V \rightarrow \Re$ be a convex and concave functions (respectively) on a convex set $\Omega \in V$, then the sets*

- $\{\omega \in \Omega | f_1(\omega) \leq b\}$, *and*
- $\{\omega \in \Omega | f_2(\omega) \geq b\}$

are convex for any $b \in \Re$.

ANTENNA ARRAY WEIGHT DESIGN CONVEXITY

Basic Theorems

Theorem 1 *Let* $\{f_1, f_2, \ldots, f_n\}$, *be a set of real valued convex functions on a convex set* $\Omega \subseteq V$, *then the following functions are convex on* Ω.

- $F_1(x) = \max\limits_{1 \le i \le n} f_i(x)$.
- $F_2(x) = \Sigma_{i=1}^{n} \alpha_i f_i(x)$, *where* $\alpha_i \geq 0$.

Theorem 2 *Let* $f : \Sigma \times \Omega \rightarrow \Re$, *be a real valued function, where* $\Omega \subseteq V$ *is a convex set and* $\Sigma \subseteq U$ *is any subset of* U. *If the function* $f_\sigma(\omega) = f(\sigma, \omega)$ *is convex on* Ω *for any fixed* $\sigma \in \Sigma$, *then the function* $F : \Omega \rightarrow \Re$ *defined by*

$$F(\omega) = \sup_{\sigma \in \Sigma} f(\sigma, \omega)$$

is convex on Ω.

ANTENNA ARRAY WEIGHT DESIGN CONVEXITY

Some Convex Functions of Weight Vector

- Total noise power, $P_n = w^* \Sigma w$.
- Real and imaginary parts of a certain (scaled) element weight, αw_i.
- Magnitude of a certain (scaled) element weight, αw_i.
- Real and imaginary parts of scaled noise-free pattern function, $\alpha S(\theta, \omega)$.
- Scaled array noise-free pattern gain, $|\alpha S(\theta, \omega)|$.
- $\max_{\omega_1 \leq \omega \leq \omega_2} |S(\theta, \omega)|$ for any fixed θ.
- $\max_{\theta_1 \leq \theta \leq \theta_2} |S(\theta, \omega)|$ for any fixed ω.
- $\max_{\theta_1 \leq \theta \leq \theta_2, \omega_1 \leq \omega \leq \omega_2} |S(\theta, \omega)|$ are also convex for any $\theta_1, \theta_2, \omega_1$, and ω_2.
- Any positive combination of the above functions.

ANTENNA ARRAY WEIGHT DESIGN CONVEXITY

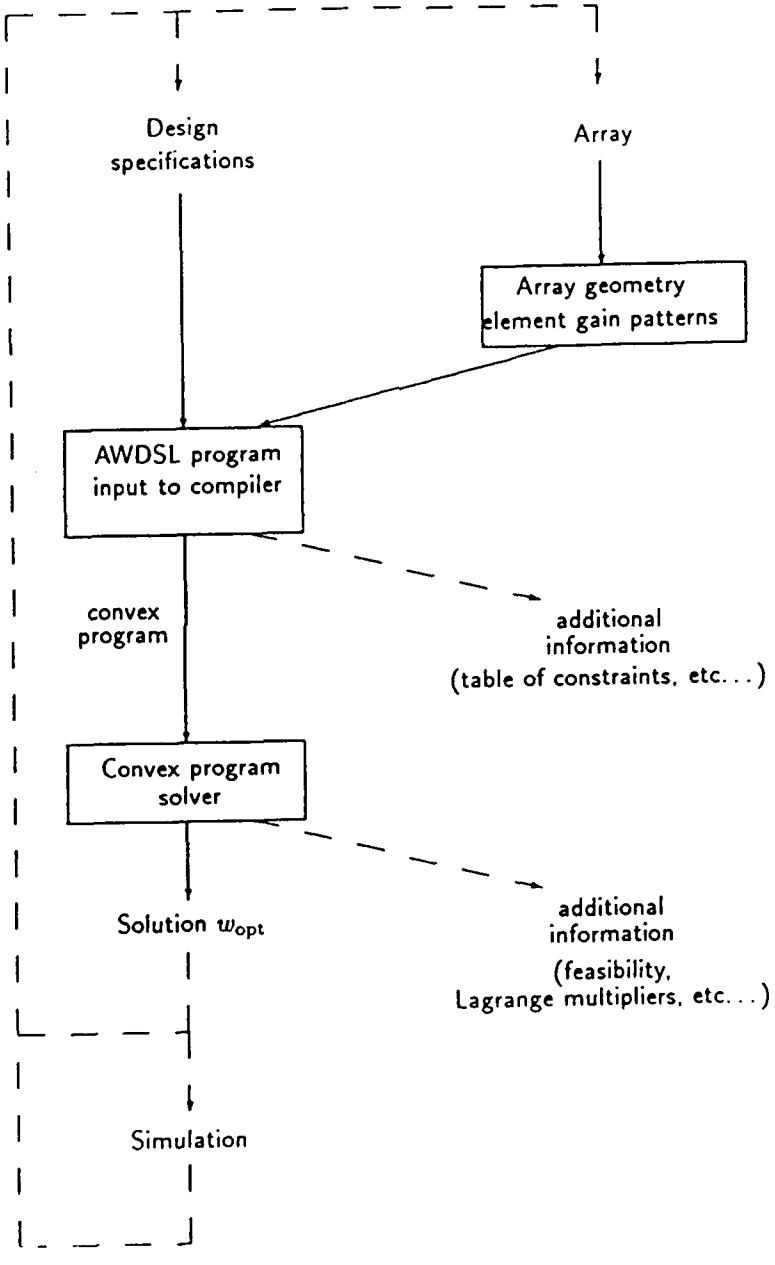
Why Convexity

- Efficient algorithms for solving convex programs exist. That is, such algorithms
  - are fast,
  - are numerically stable,
  - can detect whether an optimal solution exists or not.
- Locally optimal solutions are globally optimal.
- Additional information, such as the Lagrange multipliers (dual prices) are extremely useful in judging a design.

ANTENNA ARRAY WEIGHT DESIGN
GENERAL CONVEX WEIGHT DESIGN

General Outline

- Reduce weight design problems to convex programs.
- Approximate general convex programs by ones for which efficient algorithms exist.
- Solve approximate convex programs.
- Report optimal weights and shadow prices (Lagrange Multipliers).


Design specifications
Array
Array geometry element gain patterns
AWDSL program input to compiler
convex program
additional information (table of constraints, etc...)
Convex program solver
Solution w_{opt}
additional information (feasibility, Lagrange multipliers, etc...)
Simulation

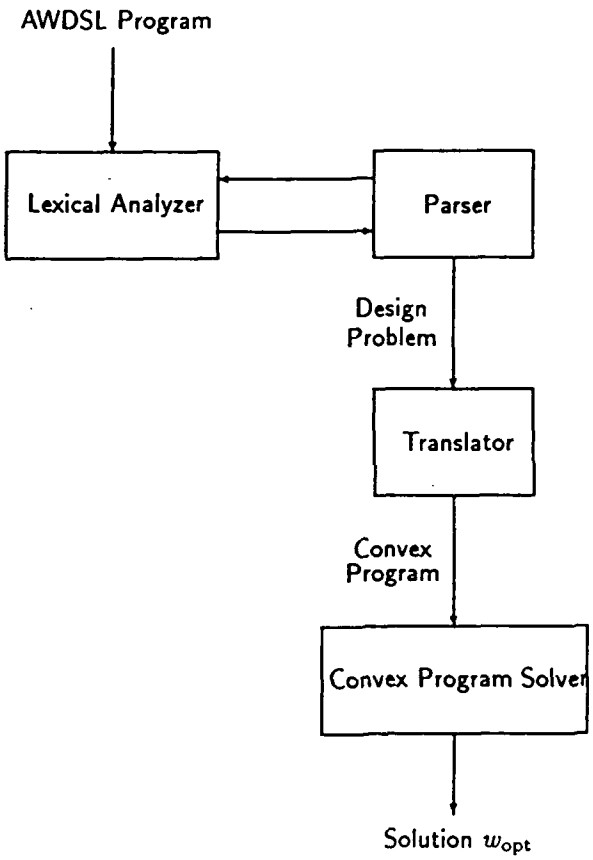


Figure 1: Our implementation of awd, a specification language compiler and solver.

ANTENNA ARRAY WEIGHT DESIGN
GENERAL CONVEX WEIGHT DESIGN

Questions and Answers

Question: What antenna weight design problems can be reduced to convex programs?

Answer: A very general class of weight design problems can be cast as convex programs.

Question: Can large problems be translated to convex programs in real time?

Answer: Automating the translation step from a design problem to a convex program using computers rather than having it manually done, makes it possible to handle large problems.

Question: How closely does the approximate program represent the original problem?

Answer: If the translation is accurately done, the approximate program can be made arbitrarily close to the original problem.

ANTENNA ARRAY WEIGHT DESIGN
GENERAL CONVEX WEIGHT DESIGN

Questions and Answers

Question: Can the approximate program be solved effectively and are such solutions reliable?

Answer: If the approximate program is a linear or quadratic program, then there are fast, reliable and effective algorithms to find solutions.

Question: Can the mathematical program solution be related to the original design problem?

Answer: An *inverse translation* is necessary to relate the optimization results to the original problem.

Question: Are there any extra advantages that warrant the extra effort?

Answer: Additional information available at the end of the optimization (e.g. Lagrange multipliers) is as important as the actual solutions.

Also, if such method fails, then no other method can succeed in solving the problem. Non-convex iterative methods cannot assert such a powerful claim.

Question: Where does the proposed method fit in the typical *engineering design* cycle?

Answer: To be discussed later.

A Proposed Approach To Convex Design

- Introduce a specialized specification language, called *Antenna Weight Design Specification Language* or *AWDSL*, to facilitate the specification of a convex weight design problem.
- Introduce a specification language *compiler* to
  - automate translating a design problem into a convex mathematical program, and
  - link the solutions and whatever extra information resulting from optimization to the original design problem.
- Choose a class of convex optimization problems which can arbitrarily approximate the original design problem and for which effective algorithms exist.
- To facilitate implementing optimization techniques, think of a *complex* weight vector as a pair of *real* vectors.

$$w = w_{\Re} + jw_{\Im}$$

where

$$w_{\Re}, w_{\Im} \in \Re^N$$

are the real and imaginary components of the complex weight vector.

Our Proposed AWDSL Structure

```
array_description {
   description of array geometry
   and element characteristics
          ...
}
minimize {
   design objectives
          ...
}
subject_to {
   design constraints
          ...
}
```

The *Array Description* Section

```
array_description{
/* number of antenna elements in array */
   n_elements = number of elements;

/* lower and upper bounds on frequency band *
   lower_freq = lower bound;
   upper_freq = upper bound;

/* specify the x locations */
   element_x_loc(i) = x-loc of element i;

/* specify the y locations */
   element_y_loc(i) = y-loc of element i;

/* specify the element noise powers */
   element_pwr(i) = power of element i;

/* specify the initial weights */
   element_w(i) = (real part , imaginary part);
}
```

The *Objective* Section

An objective term is of the form

expr * *g*;

where *expr* is any positive expression and g is one of the following convex functionals

- `noise_pwr`,
- `max_mag_S`(θ_1, θ_2, j),
- `max_mag_S`$(\theta_1, \theta_2, j)(f)$, or
- `max_mag_S`$(\theta_1, \theta_2, j)(f_1, f_2, k)$.

The *Constraints* Section

Constraints are in the form of *functional inequalities* such as

g <= *expr* ;
expr >= *g* ;
g >= *expr* ;
expr <= *g* ;
g == *expr* ;
expr == *g* ;
expr1 <= *g* <= *expr2* ;
expr1 >= *g* >= *expr2* ;
abs(*g*) <= *expr* ;
expr >= abs(*g*);

The *Constraints* Section/continued

The g is one of the following functionals

- re(*scl_fctr*∗W(i)),
- im(*scl_fctr*∗W(i)),
- mag(*scl_fctr*∗W(i)),
- null(θ, f),
- re(*scl_fctr*∗S(θ, f)),
- im(*scl_fctr*∗S(θ, f)),
- max_mag_S(θ_1, θ_2, j),
- max_mag_S(θ_1, θ_2, j)(f), and
- max_mag_S(θ_1, θ_2, j)(f_1, f_2, k).

Producing and reporting awd results

Optimization usually ends with

- `found strong minimum`: the unique feasible optimal weight vector (w) was found.
- `found weak minimum`: non-unique feasible weight vector (w) that minimizes the objective function was found.
- `solution appears to be unbounded`: happens only if the weight vector (w) in the AWDSL program does not have explicit bounds. It is generally a symptom of an ill-formulated design problem.
- `no feasible point found`: there is no weight vector (w) that satisfy the constraints. Some (or all) of the constraints in the *constraints* section must be relaxed.

Output Listing

The listing for design constraints will be six-column lines containing the following information

`constraint value lo_bnd up_bnd lambda status.`

These columns are as follows.

Column 1: description of functional, e.g.

- `re(`*scl_fctr*`*W(`i`))`, or may be
- `max_mag_S(`θ_1`, `θ_2`, `j`)(`f_1`, `f_2`, `k`)`.

Column 2: value of functional.

Column 3: value of lower bound on functional.

Column 4: value of upper bound on functional.

Column 5: value of Lagrange multiplier for functional.

Column 6: This field is empty if functional is within its bounds. Otherwise, it will contain one of:

- `lb`
- `ub`
- `violates lb`
- `violates ub`

Sample Ouput Listing

The following segment in the *constraints* section of the source file *filename*

```
re(S(0)) >= 1.0;
mag(S(0)) <= 1.5;
null(270,0.9);
```

may produce the following lsiting

```
re(S(0))          1       1    1e+10   1.04 lb
mag(S(0))         1.082   0    1.5     0
null(270,0.9)     0       0    0       0
```

Similarly,

```
max_mag_S(90,270,50)(1.1) <= 0.1;
```

may produce the following lines in *filename*.out

```
max_mag_S(90,270,50) 0.1035   0   0.1   1.87   ub
        Above constraint attains bounds at:
          1.1      90        0.1035      1.24
          1.1      180       0.0999      0.63
```

Prometheus Inc.
Final Report
11 July 1990

Section VIII
A Specification Language

A Specification Language Approach to Solving Convex Antenna Weight Design Problems

Nasser Khraishi, Richard Roy and Stephen Boyd[1]
Prometheus Inc.
103 Mansfield St.
Sharon, MA 02067
Technical Report #90-01

April, 1990

Abstract

In this paper we introduce a new computer-aided approach to the design of weight vectors for antenna arrays of omnivariant elements responding to signals in the far-field of the array. The problem of concern is the deterministic problem, where frequencies and directions of arrival for sources are assumed to be known.

The approach relies on the ability to cast many typical design problems as convex programming problems with the real and imaginary parts of the complex weight vector as the decision variables. In this approach we introduce an *antenna weight design specification language* compiler as a tool for translating design objectives and specifications from everyday engineering language into a convex mathematical program. Resulting convex program is then passed to a standard convex program solver and results are reported in an easily interpretable form.

1 Introduction

1.1 Design of Directivity Patterns

The problem of finding array element weights so as to shape certain directivity patterns for the sensor array is an old one and the approaches used are quite diverse. In general, we can divide the methods used into two main categories, closed form and iterative techniques. On the one hand, closed form techniques are easy to utilize, yet the class of problems for which such methods can be used are usually restricted. Thus, the computational efficiency

[1]The authors are currently with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305

is offset by the limited applicability. On the other hand, iterative methods usually attain the less restricted applicability by sacrificing the computational ease offered by closed form solutions.

Historically, closed form methods were the first to appear in literature. The ability to use tables, charts, and hand calculations were, in our opinion, the real reason for this. For a transmitting array of equally spaced elements, Stone[1] used a binomial expansion representation of the array pattern with the exciting currents as the binomial coefficients. Schelkunoff [Sch43] used the ability to represent arrays of equally spaced antenna elements as polynomials to decide on the appropriate currents to shape directivity patterns of transmitting antenna arrays by intelligent choice of zeroes of the representing polynomial.[2] Dolph [Dol46, Rib47, Pri53] used Tchebychev polynomials to find an exceedingly easy solution for the optimal tradeoff between sidelobe gain and mainlobe width. Dolph initially assumed a linear array of equally spaced narrowband transmitting elements symmetrically placed around the origin so that the combined array output can be assumed totally real, and that the exciting currents are in-phase. Different attempts were made afterwards to relax some or all of these assumptions.

The significance of Dolph's work is that he was the first to talk about a solution to the pattern shaping problem that is *optimal* in some sense. This in itself differentiates his work from earlier ones which gave *a* (and not *the*) solution to the problem. What followed was a search for solutions that are optimal in one sense or another. For instance, one of the methods proposed was that of representing array directivity as a ratio of two "Hermitian" forms and attempting to maximize this ratio by use of generalized eigenvalues and eigenvectors arguments [CT65, TC67]. Attempts to handle extra constraints on the pattern can be found in [SB71, VB72, Che71]. The last two papers have a good review of the literature (Western and Eastern) on this approach. Such Hermitian ratios are not *convex*, and thus solutions obtained for the constrained case are generally *local*. That is, it is almost always the case that there is a better solution that the method did not locate.

Waren *et. al.* [LSW66, WLS67], suggested the use of mathematical programming for optimizing different aspects of antenna design (including geometry). The problems that were posed, whether weight design or otherwise, were generally non-convex. Lack of convexity meant that produced solutions were typically "local," and that the failure to produce a set of parameters that satisfy the design specifications did not answer the question about the solvability of such a problem. That is, the method failed to answer the question of whether there exists any set of parameters for which the design specifications can be satisfied. Furthermore, convexity is an underlying assumption for nearly all efficient optimization algorithms. Thus, specifying non-convex optimization problems limits the ability to utilize most of the existing efficient optimization procedures.

Of the different methods utilizing convex programming for antenna array weight design,

[1] U. S. Patents 1,643,323 and 1,715,433

[2] Using the principle of reciprocity [Kra88], exciting currents for transmitting antennas can be thought of as element weights for receiving antennas.

we choose to mention those of [MHM72, EF75, Wil76, Wil78]. Evans and Fortmann [EF75] studied the related problem of optimizing patterns for antennas with continuous line-source excitation. A functional analysis argument was used to solve the problem. Although this problem is removed from our discussion, the link can be seen from the fact that an array can be thought of as a "sampled aperture" (see for example [Ste76]). Of particular interest to us are the works of Wilson [Wil76, Wil78] and McMahon, *et. al.*, [MHM72]. In particular, the optimization method we utilize is similar to that proposed in [MHM72].

In [MHM72], the constraints on the magnitude of the pattern were translated to constraints on the real and imaginary parts of the pattern function (with a possible 3dB error). The real and imaginary parts of the weights were taken as the design variables. Thus, constraints on the pattern were translated to linear constraints on the design variables. In order to obtain a linear programming solution of the above problem, the authors "created" a linear objective function especially tuned so as to reduce the mainlobe beamwidth. One interesting aspect of the above approach is that the authors, although their concluding discussion shows that they appreciated the significance and generality of their approach, in our opinion failed recognize a major advantage that their approach had over earlier ones. Since the problem solved was, due to discretization of the pattern and the 3dB errors, less restrictive than the original problem (where the magnitude of the pattern strictly lies within the designated limits for all angle points), and since the original problem was a convex one, it is obvious that if the method discussed above failed then there are no element weights that will satisfy the original constraints. Thus, if the designer, using such a method, failed in obtaining solutions satisfying the constraints, the problem itself as posed should be re-examined rather than attempting different approaches to solving the problem.

Wilson, [Wil76, Wil78], utilized a similar method. In his approach, Wilson described the array pattern as a function of the phase and magnitude of element weights. By imposing symmetry of the array around the origin, the phase of the weights was assumed to be constant and the pattern was assumed to be totally real (not unlike Dolph's assumptions). Thus, the only variables remaining were magnitudes of the weights. An approximation scheme, [BY66], was then used to find the variables (magnitudes of weights) that will minimize the Tchebychev (infinity) error between the actual response and a desired response at selected angle points. This approximation scheme is based on linear programming techniques. The interesting note about this method is that had Wilson assumed the decision variables to be the real and imaginary parts of the weights, rather than the magnitude and phase, the symmetry constraints imposed on the problem could have been easily discarded of, and the method would have become similar to that of [MHM72].

1.2 A Proposed Approach to Optimal Directive Pattern Shaping

In this paper we propose a new approach to optimal directive pattern shaping. We only concern ourselves with design constraints and objectives that are proven to be convex

functionals of the weight vector. Non-convex problems will not be considered. This is not to say that such problems are not important. For example, the problem of finding the optimal geometry to attain a certain pattern is a very interesting one. Another interesting non-convex design problem is considered in [LJ88]. In that work the authors study the problem of minimizing the number of active elements (or maximizing the number of zero elements) in the antenna array while satisfying certain design constraints.

Unfortunately, the current status of mathematical programming does not allow handling such problems efficiently. Moreover, solutions produced with existing algorithms will, at best, be local solutions. That is, there is no way of knowing how far such a solution is from the global optimum one. Furthermore, failure of such methods to produce an initial feasible point (if the constraints that define the set of feasible designs are not convex) does not convey much information about the *existence* of a feasible point (i.e. one which satisfy design constraints).

In addition, experience from other fields, such as economics, indicates that the extra information, e.g. the Lagrange multipliers or shadow prices as they are often referred to, available from a typical convex program solver, is in many instances as important as the actual solution itself. These shadow prices reflect to the designer the tradeoff between the "tightness" of the design constraints and the value of the design objective, thus, giving the designer an opportunity to decide whether keeping the constraints as tight as they are is "worth it."

We also propose the use of a specialized design language compiler, referred to as the *antenna weight design specification language*, as a mechanism for translating the design problem from engineering specifications to the corresponding convex mathematical program. The significance of this approach is that it automates the important step of translating antenna weight design problems into convex mathematical programming ones. The antenna designer need not be an expert in mathematical programming. Thus, the designer should not carry the burden of translating the engineering specifications, which may resemble a large number of objectives and constraints, into a convex mathematical program. The designer should also expect results of the design process to "make sense" from an engineering point of view. Although mathematical programming results from a typical convex program solver contain enormous information, this information need not make sense to a mathematical programming novice.

This specification language approach, combined with convex programming, has been successfully used recently as a computer aided design tool in designing controllers [BBB*88, Khr90] for linear time-invariant dynamic systems. In that work, the ability to *parameterize* all stabilizing controllers of a dynamic system combined with the fact that many specifications on the control system translate into convex constraints on these free parameters, were exploited to represent the design problem as a convex program.

2 The Antenna Weight Design Problem

2.1 Underlying Assumptions

Consider a planar array of N sensor elements with fixed relative locations. Elements are assumed to have negligible gain outside a predetermined (finite) frequency band of operation. This can be attained in practice by passing the sensor output through an "ideal" band-pass filter (BPF) before processing the signal. Medium of transmission is assumed to be isotropic and non-dispersive. Signal sources are assumed to be farfield and lie in the plane of the array as shown in Figure 1. Furthermore, it is assumed that each sensor in the array will have an additive zero-mean stationary noise component at its output. This noise can be due to thermal, amplifier, and/or spatial noise. This noise is assumed to be uncorrelated with the signal component. Without loss of generality, it is also assumed that noise components in different elements are uncorrelated.[3]

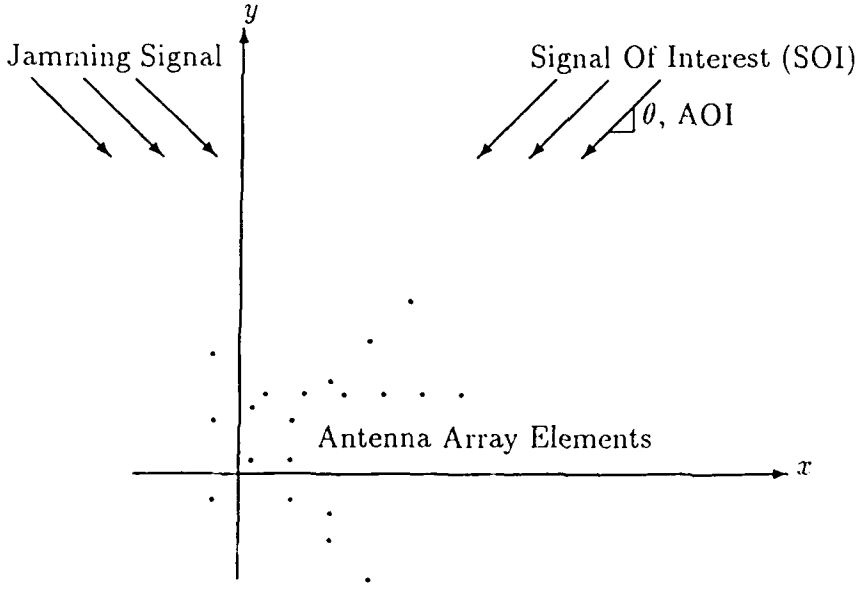


Figure 1: Antenna array under consideration.

Letting $s_i(\theta, \omega)$ be the output of the i^{th} element with respect to a certain signal arriving from the θ direction with angular frequency ω, a fundamental assumption is that the

[3]A simple *whitening* argument can be used to justify this for correlated noise.

combined array output due to such signal can be written as

$$y(\theta,\omega) = \sum_{i=1}^{N} \bar{w}_i(s_i(\theta,\omega) + n_i). \tag{1}$$

where w_i is referred to as the *element weight* of the i^{th} element, $\bar{w}_i$ is its complex conjugate, and n_i is the corresponding additive noise for such an element. This is equivalent to saying that element weights affect the combined array output in a *linear* fashion. Letting w, $s(\theta,\omega)$, and n be complex vectors in $\mathcal{C}^N$, with the i^{th} element of these vectors as w_i, $s_i(\theta,\omega)$, and n_i, respectively, then another way to state the combined array response is

$$y(\theta,\omega) = w^*(s(\theta,\omega) + n) \tag{2}$$

where the '$*$' operation represents taking the conjugate transpose (Hermitian) of the vector w. This w is usually referred to as the *weight vector*. Sometimes we shall refer to the combined array noise-free response as

$$S(\theta,\omega) = w^* s(\theta,\omega). \tag{3}$$

As for the noise component, under the assumptions above, the total noise power in the output can be thought of as

$$P_n = \sum_{i=1}^{N} \sigma_i |w_i|^2, \tag{4}$$

where,

$$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N) = E(nn^*). \tag{5}$$

i.e. the σ's are non-negative real numbers.

The relationship between sensor element attributes and the combined array output is shown in Figure 2 for a linear array of N elements. In this figure we have, on purpose, distinguished between the noise-free response of each sensor element, s_i in the above discussion, and the additive noise, n_i.

2.2 Problem Statement

Assume that we have an antenna array as described above and that the array is receiving signals with potential interference or jamming. Using adaptive antenna design techniques, a certain criterion about the quality of reception of a desired signal versus reception of the jamming, such as SNR, can be optimized. Most such methods react to jamming by placing a pattern *null* (that is a zero of the directive pattern function) at the directions in which jamming is expected. Although these solutions are optimal as far as the optimization criterion is concerned, they are quite sensitive to changes in incident angles and other deviations from underlying assumptions.

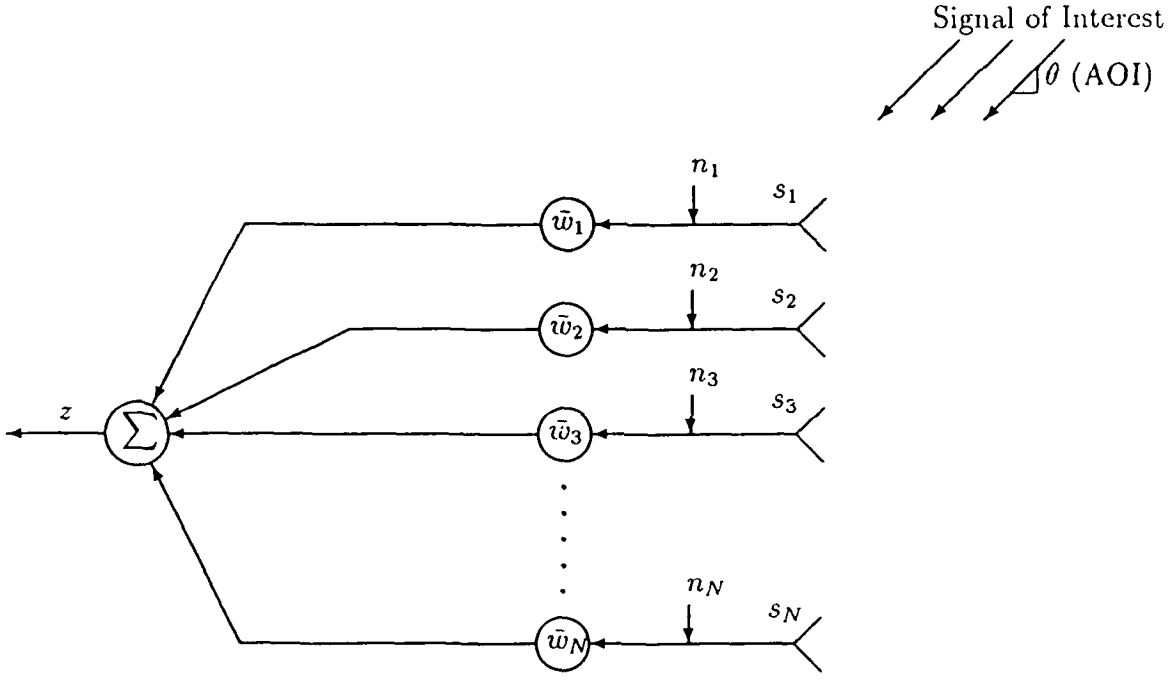


Figure 2: Combined array output as a function of the weight vector, element noise-free responses and additive noise.

For the deterministic weight design problem, it is assumed that desired signals are arriving at known angles of incidence with prespecified frequencies. It is also assumed that jamming signals are arriving at predetermined angles and frequencies. The weight design problem can be stated as follows.

> Find an antenna array (complex) weight vector that extremizes a given design objective while satisfying certain design constraints on the directive pattern function of the array.

3 The Convex Mathematical Programming Approach

3.1 Basic Definitions

In this section, some of the basic definitions and terminology used in the remainder of this paper will be reviewed. For a detailed discussion, the reader is advised to refer to any standard textbook on the subject (e.g. Luenberger [Lue69, Lue84], Bazaraa and Shetty [BS79], or Rockafeller [Roc70]).

In the ongoing argument, V is assumed to be any linear space on the real field $\Re$.

Definition 1 (Convex Set) *$\Omega \subseteq V$ is said to be a convex set if $\mu x + (1-\mu)y \in \Omega$ whenever $x, y \in \Omega$ and $0 \leq \mu \leq 1$.*

This is to say that if two points are in a convex set, then any point between them on the line connecting the two points is also in the set. It is usually assumed that the empty set is a convex set.

Definition 2 (Convex Function) *A function $f : V \rightarrow \Re$ is a convex function on a convex set $\Omega \subseteq V$ if $f(\mu x + (1-\mu)y) \leq \mu f(x) + (1-\mu)f(y)$ for all $x, y \in \Omega$ and $0 \leq \mu \leq 1$.*

This is to say that the line connecting two points on the graph of a convex function lies above the graph of the function between these two points.

A convex mathematical program is the problem of finding an element in a given convex set that minimizes the value of a certain convex function on the set. This can be stated as

$$\begin{array}{ll} \text{minimize} & \Phi(\omega) \\ \text{subject to} & \omega \in \Omega \end{array} \tag{6}$$

where $\Phi : V \rightarrow \Re$ is a convex function on the convex set $\Omega \subseteq V$. This convex function, Φ, is usually referred to as the *objective function* while the set Ω is often referred to as the *feasible set*. Ω is usually assumed to be a closed set.

Many of the results on convexity will not be mentioned here, but it should be noted that an intersection of convex sets is convex, and that any positively weighted linear combination of convex functions is a convex function.

3.2 Why Convex Programming

A general mathematical programming problem is the same as that described in 6, except for (one of) the convexity requirements on the objective function and the set. Although this provides a more general setup that allows formulating much more general design problems, such as those mentioned in the introduction, yet, it is desired to maintain convexity in the mathematical programming formulation of the problem for the reasons discussed below.

The most important feature that a convex program has is that local optimal solutions are global optimal solutions. That is, no local solutions exist. Thus, the notion of global optimality for a convex program is well defined. That is, it can be easily verified whether a solution is the optimal one or not by using some of the well-known local optimality tests, such as the Kuhn–Tucker conditions [Lue84]. This is not true for a general mathematical program since, generally speaking, there is no quick test for global optimality of a solution since solutions for a general mathematical programming problem can be local. Furthermore, local optimal solutions for a general non–convex program may be arbitrarily far from the global optimal one. Thus, it may very well be the case that starting the same algorithm from different initial points will produce different results.

Another advantage that convex programming has over general mathematical programming is the abundance of convex programming algorithms that are *effective* and *robust*. By effectiveness it is meant that given a convex programming problem of a certain type, an algorithm can be found that will correctly determine whether there are any feasible points. If there are any, the optimal one will be found. On the other hand, robustness means that an algorithm is numerically stable and will handle a wide spectrum of problems of a certain type (*e.g.* linear programs) while producing results that are correct within a prespecified numerical tolerance. Furthermore, for most classes of convex programming problems, algorithms have been developed with well estimated rates of convergence. Thus, such algorithms are guaranteed to produce results within a prespecified precision in a precalculable amount of time.

One more useful feature of convex programs is that solution sets are convex. Thus, even for non-differentiable convex optimization problems [Kel60, Sho85, Akg84], there are many methods that will try to find the solution set by continuously refining a convex set that contains the solution set. Many of these methods have well established (although generally poor) rates of convergence. Many of these methods are still undergoing intensive research which may have a promise for the future.

Finally, most algorithms which solve convex programs will produce what is known as the Lagrange multipliers or shadow prices which resemble a solution for a *dual* optimization problem. Although the main use of these multipliers within an algorithm is to verify the optimality of the solution found, they have another interesting practical interpretation. These multipliers reflect the inherent tradeoff among objectives and constraints. That is, the multipliers (shadow prices) tend to quantify the answer to the question

"how much does maintaining a constraint as tight as it currently is contribute to the value of the objective function?"

A large positive multiplier indicates that relaxing the lower bound on the related constraint may significantly reduce the value of the objective function, while a large negative multiplier indicates that the upper bound should be examined.

This is quite an important feature since in most engineering applications and otherwise, the designer does not usually have an *a priori* knowledge of the exact tradeoffs among certain design constraints and design objectives. Quantifying these tradeoffs is a valuable asset to the designer.

3.3 The Antenna Weight Design Problem as a Convex Program

The ability to write a weight design problem as a convex mathematical program is now investigated. In particular, we note that there is a one-to-one correspondence between vectors of $\mathcal{C}^N$ and vectors in $\Re^{2N}$. For a vector $w \in \mathcal{C}^N$ can be thought of as $w_\Re + jw_\Im$, where $w_\Re, w_\Im \in \Re^N$ and $j = \sqrt{-1}$. Similarly, we can think of a vector $W \in \Re^{2N}$ as $W^T = [w_\Re^T, w_\Im^T]^T$. That is, the first N elements of a vector in $\Re^{2N}$ can be thought of as corresponding to the real part of a complex vector in $\mathcal{C}^N$, while the latter N elements can be thought of as corresponding to the imaginary part of the complex vector. This remark permits the study of the weight design problem as a problem on $\Re^{2N}$, where N is the number of antenna elements, rather than a problem on $\mathcal{C}^N$.

In the sequel, it will be assumed that the weight vector $w \in \mathcal{C}^N$ is written as $w = w_\Re + jw_\Im$, where $w_\Re, w_\Im \in \Re^N$ are called the real and imaginary components of the weight vector. Instead of explicitly considering $W^T = [w_\Re^T, w_\Im^T]^T$ as a vector in $\Re^{2N}$, the pair $w_\Re$ and $w_\Im$ will be considered.

The relation between $w_\Re, w_\Im$ and W is a simple linear transformation of the form

$$w_\Re = [I_N | 0_N] W, \text{ and}$$
$$w_\Im = [0_N | I_N] W,$$

where I_N is the $N \times N$ identity matrix, while 0_N is the $N \times N$ zero matrix. Thus, a linear function on $w_\Re$ and $w_\Im$ corresponds to a linear function on W, and a convex function on $w_\Re$ and $w_\Im$ corresponds to a convex function on W.

With the above remarks in mind, some convex functions on the real and imaginary parts of the weight vector are discussed next. Such functions may appear as an objective or be used to construct constraints by setting upper and/or lower bounds. Here, it should be noted that, to maintain convexity of the set satisfying the constraints, lower bounds can be set only for linear functionals.

- Total noise power, $P_n = \sum_{i=1}^N \sigma_i |w_i|^2 = w^* \Sigma w$, can be rewritten $P_n = w_\Re^T \Sigma w_\Re + w_\Im^T \Sigma w_\Im$, or $P_n = \sum_{i=1}^N \sigma_i (w_\Re^2 + w_\Im^2)$. This is obviously a convex function since the individual σ's are non-negative.

- The real and imaginary parts of a certain element weight, w_i, are (obviously) linear functions of the real and imaginary parts of the weight vector.
- The magnitude of a certain element weight, w_i, is (obviously) convex in the real and imaginary parts of the weight vector.
- Any norm of the weight vector, is (obviously) convex in the real and imaginary parts of the weight vector.
- The real and imaginary parts of the noise-free pattern function, $S(\theta, f)$, are linear in the real and imaginary parts of the weight vector. To see this, note that $S(\theta, f) = w^* s(\theta, f)$; letting $s(\theta, f) = s_{\Re}(\theta, f) + j s_{\Im}(\theta, f)$, it becomes obvious that the real part of S is $w_{\Re}^T s_{\Re} - w_{\Im}^T s_{\Im}$, while the imaginary part of S is $w_{\Re}^T s_{\Im} + w_{\Im}^T s_{\Re}$, both of which are linear.
- The array noise-free pattern gain, $|S(\theta, f)|$, is convex in the real and imaginary parts of the weight vector. This can be easily seen from the fact (the previous point) that the real and imaginary parts of $S(\theta, f)$ are linear in the real and imaginary parts of the weight vector.
- Also, $\sup_{f_1 \le f \le f_2} |S(\theta, f)|$ for any fixed θ, $\sup_{\theta_1 \le \theta \le \theta_2} |S(\theta, f)|$ for any fixed f, and $\sup_{\theta_1 \le \theta \le \theta_2 f_1 \le f \le f_2} |S(\theta, f)|$ are also convex for any θ_1, θ_2, f_1, and f_2.
- Finally, any positively weighted linear combination of the above functions is also convex.

4 The Specification Language Approach

It was shown above that many typical antenna weight design problems for a large class of antennas can be posed as convex mathematical programs. The *theoretical* value of this approach was discussed. The practicality of such a design method remains limited, as far as the antenna weight designer is concerned, unless such a designer can specify the actual design problem as a convex program and interpret the results efficiently. A *computer aided design* approach to *translating* the antenna weight design problem discussed above into a mathematical program is a most natural way for solving this dilemma.

To do this, we propose the use of an *antenna weight design specification language (AWDSL)*. The purpose of such a language is to enable the designer to fully specify the design problem in everyday engineering language and not as a convex mathematical program. Thus, the designer can set up the design problem in a natural way by, say, setting pattern nulls, specifying limits on the magnitude of the array response, or selecting any of the convex functionals mentioned above as a design objective. A specification language *compiler* then translates the design problem into a convex mathematical program. The designer need not

know the resulting mathematical program. This is not unlike the fact that a good computer programmer need not know anything about *the machine language* for the computer that is being used without sacrificing the ability to produce correct and efficient computer programs. This convex program can then be solved using any convex program solver. Afterwards, the results are reported to the designer in a form similar to what was used to specify the problem. This general idea is outlined in Figure 3. Using the above outline, a description of the

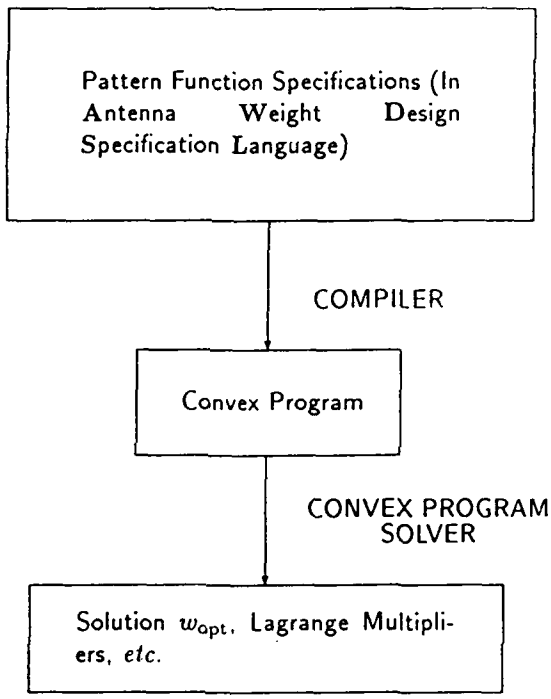


Figure 3: Outline of how to implement a specification language approach.

specific approach utilized in this paper, called awd [Khr88], is illustrated in Figure 4

Figure 5, shows how this specification language approach fits into the typical engineering design cycle. The designer starts to design element weights for a *sensor array* so as to satisfy certain engineering *design specifications*. The designer may not know *a priori* whether the design constraints are attainable, or it may be the case that the current specifications are based on earlier iterations (the dashed line in the Figure). Using the array and design specifications, the designer sets up an *AWDSL program* and inputs it to the language compiler resulting in a *convex program*. Additional information about the problem may be available

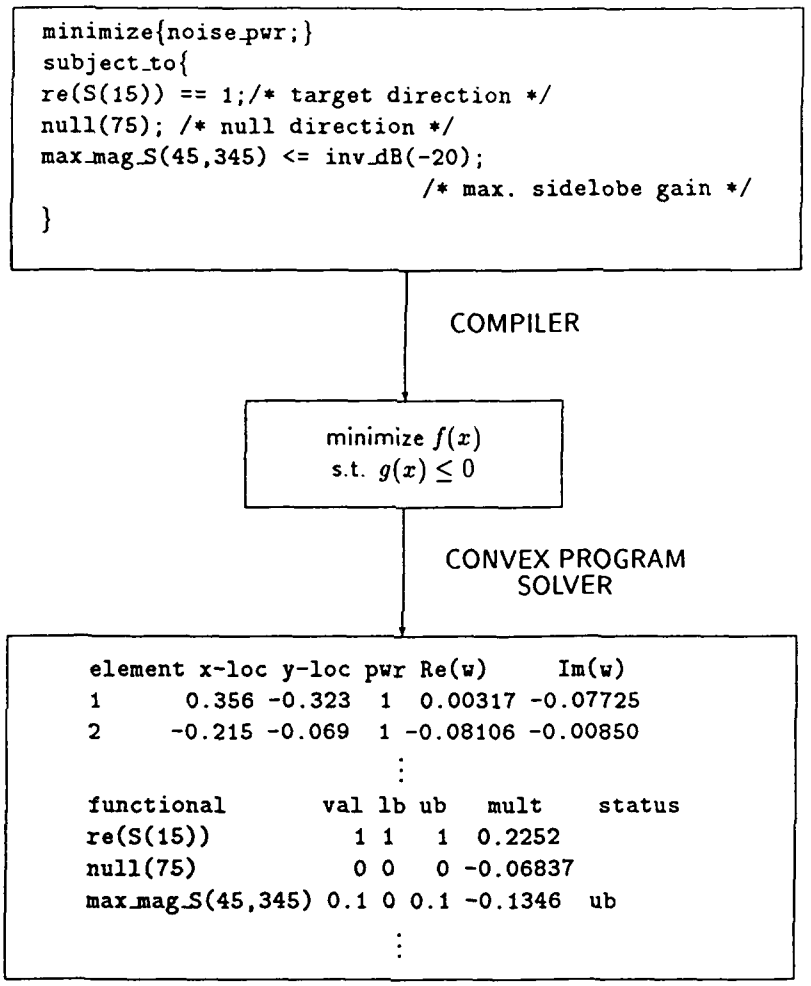


Figure 4: How awd fits in terms of the general specification language approach.

too. The convex program is then passed to a *convex program solver*. The designed weights (w_{opt}) and certain additional information (e.g. Lagrange multipliers) will be available at the end of this phase. If the problem is feasible, simulating the results will show whether they are implementable or not. Results of the simulation, combined with the information about problem feasibility and the Lagrange multipliers, may serve as a tool for refining the design and/or array specifications.

Infeasibility of the problem (under convexity) gives a clear answer to whether the design specifications can be attained or not by any set of weights. If the problem is infeasible, the answer is an unqualified "no" as was discussed earlier. This in itself is a valuable product of the method. For if the problem is infeasible, the designer should decide, from the additional information provided by the convex program solver, which design constraints were the infeasible ones. Then these "violated" constraints should either be "relaxed," if this is acceptable, or a different array be used if it is vital to maintain the original constraints. By a different array we mean that the geometry and/or gain patterns of individual sensors are changed. It is up to the designer to decide which action to take.

On the other hand, even with a feasible problem, the Lagrange multipliers reveal to the designer the tradeoff between attaining certain design constraints and the increase of the objective value. It is up to the designer to decide whether maintaining constraints as tight as they are is worth the corresponding increase in the objective value. If the answer is no, the constraints that "cost" most to maintain should be relaxed.

From the above discussion, it can be seen that the specification language approach resembles a significant step towards making the convex programming approach to antenna weight design a practical engineering tool. Using this approach, posing the problem and interpreting the results is a task manageable by the typical antenna designer.

Of the many possible structures of an antenna weight design specification language (AWDSL), we propose the following. The AWDSL program should contain three major sections,

- *array description*,
- *objective function*, and
- *design constraints*.

The array description section is to provide all the information necessary to evaluate the objective and constraint functionals. For instance, the number of array elements and their geometry, operational frequency band, signal and noise covariance matrices, and element gain patterns should be stated in this section explicitly or by giving names of files that contain such information.

The objective section should contain design objectives. These should be convex functionals to be minimized. Positively weighted linear combinations of convex functionals and the

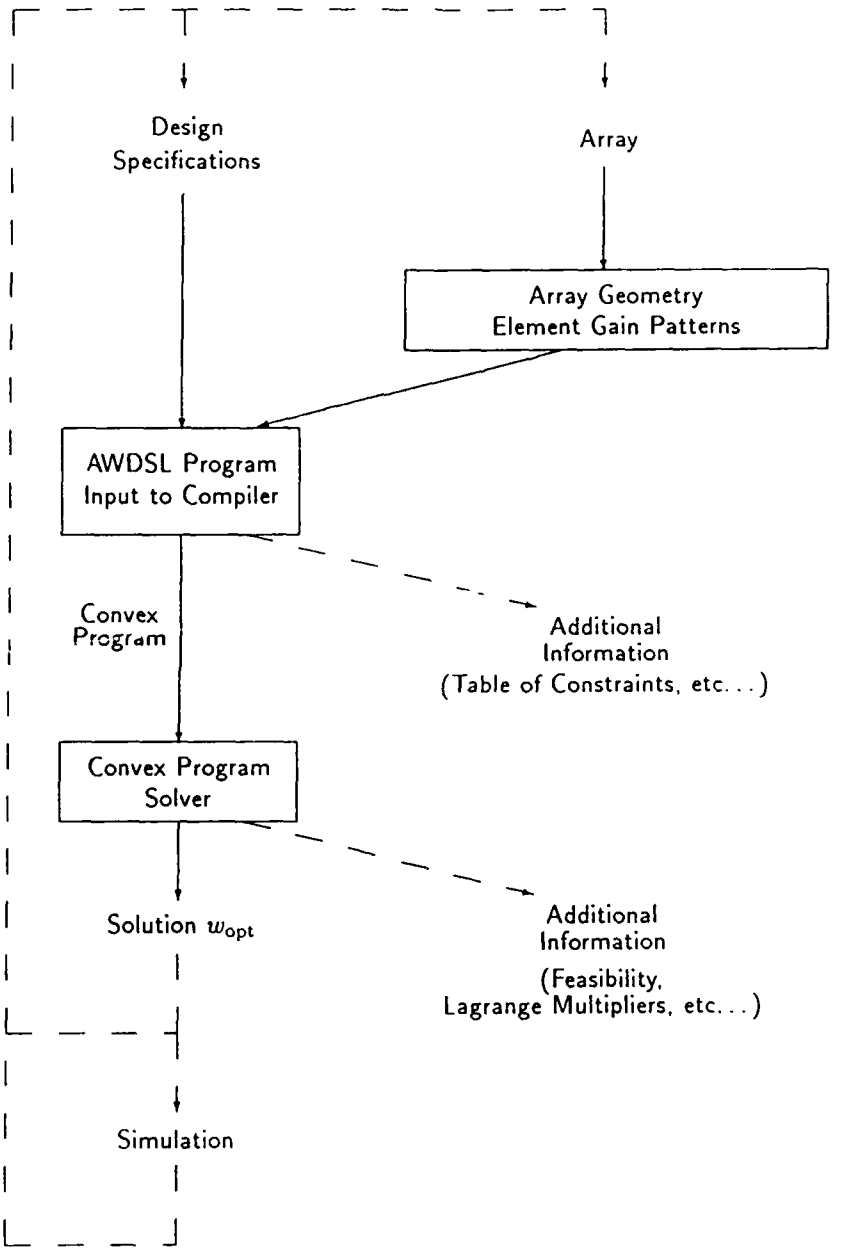


Figure 5: Overall structure of design procedure.

maximum possible value of a set of functionals are themselves convex, as was stated above, and thus may be allowed.

Finally, the constraints section should contain design constraints. Lower and upper bounds on affine functionals should be allowed. Non-affine convex functionals should have upper bounds only.

In the following we describe our implementation of **awd** [Khr88], a compiler for a small but effective subset of AWDSL.

5 An Implementation of a Specification Language

5.1 Issues Concerning Implementation

First of all we discuss some of the issues involved in choosing and implementing the AWDSL subset. In particular, the following closely related issues are involved in such an implementation.

5.1.1 Underlying assumptions

In order to simplify the problem from the computational point of view, the following assumptions are made in conjunction with those specified earlier. The compiler is to handle only planar arrays of omnidirectional elements with fixed relative locations and negligible mutual interactions. Elements are assumed to have constant complex gain over a predetermined (finite) frequency band of operation. Signal sources are assumed to be farfield and lie in the plane of the array.

It was decided that the finite operational frequency band of interest be selected by setting lower (f_{lower}) and upper (f_{upper}) frequencies in the *array description* section. Antenna element responses are assumed to be constant over the selected band. Thus, there is no need for storing or computing element patterns. It was also decided that all frequencies be specified in units of a certain nominal frequency, f_{nominal}. Although choice of f_{nominal} is arbitrary, a good choice is the center frequency of the antenna element operational frequency band. A frequency of interest (f) in physical units (Hertz) is input as a normalized frequency, $\bar{f} = f/f_{\text{nominal}}$. In other words, a frequency of 0.5 specifies a frequency of half the nominal frequency.

As usual, locations (in rectangular coordinates) of array elements must be given in units of λ_{nominal}, the wavelength corresponding to the nominal operation frequency (f_{nominal}) and the underlying speed of propagation (c) in the medium ($\lambda_{\text{nominal}} = c/f_{\text{nominal}}$).

It is also assumed that the noise-free directive pattern function of concern to the designer is the *array space factor* [RWD84, Sch43]. This is a function of Angle of Incidence, θ, (normalized) signal frequency, f, complex weights, w_i's, and element locations. Under the

above assumptions, this function is given by equation 3, where in this case,

$$s_i(\theta, f) = \exp(j2\pi f(x_i \cos\theta + y_i \sin\theta)) \tag{7}$$

where

f is the normalized (dimensionless) signal frequency,
θ is the Angle of Incidence, and
x_i and y_i are the x and y-locations (in units of λ_{nominal}) of element i.

5.1.2 Language capabilities

Not all convex functionals of the real and imaginary parts of the weight vector can be incorporated within the language without paying the price from a computational point of view. This has to do with the fact that most efficient optimization codes exploit a certain problem structure, such as linearity or differentiability. This, unfortunately, limits ability to handle all types of convex objectives and constraints discussed above. Our implementation handles design problems that can effectively be approximated by linear or quadratic programs. That is, linear constraints with linear or quadratic objectives. This choice has to do with the fact that linear and quadratic convex programming are intensively researched fields with good collection of robust and effective algorithms.

The following objective functionals fit the above description; that is, they can be effectively approximated by linear or quadratic functionals (as will be discussed in a following section).

- $\max_{f_1 \le f \le f_2} \max_{\theta_1 \le \theta \le \theta_2} |S(\theta, f)|$, maximum value that the magnitude of the array space factor assumes between angles θ_1 and θ_2 for any (normalized) frequency between f_1 and f_2.
- total noise power in the combined array output.

The same can be said about the following functionals that can be used to construct design constraints.

- $\max_{f_1 \le f \le f_2} \max_{\theta_1 \le \theta \le \theta_2} |S(\theta, f)|$, same as above.
- real, imaginary, or magnitude of $\alpha S(\theta, f)$, where α is some complex weighting factor, θ is some desired Angle of Incidence, and f is a desired (normalized) frequency.
- real, imaginary, or magnitude of αw_i, where α is some complex weighting factor, and w_i is the complex weight factor for element i.

Another design constraint can be imposed by setting $S(\theta, f)$ to zero at a given Angle of Incidence, θ, for some (normalized) frequency, f. This is termed *placing a null* at θ and f.

5.1.3 Approximations

A magnitude constraint, such as $|w_i| \leq b$ or $|S(\theta, f)| \leq b$, can be viewed as a disc in the complex plane of radius b. This disc can be arbitrarily approximated by a polygon. In particular, such a disc may be approximated by an octagon to within ± 0.34dB error. This can be accomplished, say for $|S| \leq b$ and $S = S_{\Re} + jS_{\Im}$, by replacing the magnitude constraint by the following four inequalities

$$\begin{aligned} |S_{\Re}| &\leq b\sqrt{\cos\frac{\pi}{8}} \\ |S_{\Im}| &\leq b\sqrt{\cos\frac{\pi}{8}} \\ |S_{\Re} + S_{\Im}| &\leq b\sqrt{2\cos\frac{\pi}{8}} \\ |S_{\Re} - S_{\Im}| &\leq b\sqrt{2\cos\frac{\pi}{8}}. \end{aligned}$$

As for sector conditions, such as $\max|S|$ over a certain region in AOI or frequency, the region will be *sampled* first, then the magnitude functional at each sampling point will be approximated as above. The resolution of this sampling procedure can be left to the program or be entered by the designer as an optional argument. Thus, a constraint of the form $\max|S| \leq b$ over a certain region, is approximated as $|S| \leq b$ for a set of points uniformly spaced over the region. Each of these $|S| \leq b$ constraints will be approximated by four inequalities as above.

On the other hand, $\max|S|$ appearing as an objective in the form

$$\text{minimize} \quad \max|S|$$

over a certain region in AOI and/or frequency, will be handled by introducing a new variable, say z, and replacing the problem by the equivalent problem

$$\begin{aligned} &\text{minimize} \quad z \\ &\text{subject to} \quad \max|S| \leq z. \end{aligned}$$

The latter constraint can be handled as above.

5.1.4 Implementation of language compiler

The question of how to implement the language compiler, from the programming point of view, is a fundamental one. In our implementation of **awd**, standard UNIX, [KP84, SF85], utilities were used to facilitate this formidable task. In particular, *lex* [LS84], a lexical

analyzer, and *yacc*, [Joh84], a compiler compiler, were used to construct the compiler part of awd.

As shown in Figure 6, a lexical analyzer, built using lex, will detect language *tokens* and pass them to a parser (compiler), constructed using yacc. This parser recognizes certain sequences of tokens. If a sequence is complete, a certain *action* is carried out. These actions include creating and updating a certain internal structure called the *design problem*. This design problem is actually independent of the type of optimization code used. Syntax errors are detected at this phase.

As a second phase, the design problem is translated, using a *translator* written in C language, into a convex quadratic or linear program, as appropriate. This convex program, as expected, is fine tuned to the specific convex solver used. This is why we have included the convex solver as an extra third phase within **awd** (refer to Figure 6). It should be noted that this last phase is not a part of the language. Dependence of the compiler output (the convex program) on the convex solver makes incorporating the solver with the compiler a sensible thing to do.

We have also made use of the *C-preprocessor (cpp)* [KR78], which is a UNIX utility that, among other things, allows defining constants using a "#define" facility, including external files within the source file using an "#include" facility, and permits C language style comments (*i.e.* ignore text within "/*" and "*/"). The source program is passed to *cpp* before passing it to lex.

5.1.5 Choice of a convex program solver

LSSOL, [GHM*86], a convex program solver for linear, quadratic and least squares optimization, from the Systems Optimization Laboratory at Stanford University, was chosen as the convex programming code. The code is quite stable from the numerical point of view, and is quite efficient on problems of the type encountered in the case at hand. It is the case, as was discussed above, that *most* of the convex functionals encountered in the current implementation of **awd** are linear, quadratic, or can be arbitrarily (as was shown earlier) approximated by a set of quadratic or linear functionals.

Furthermore, LSSOL utilizes the active set method [GMW81]. Thus, LSSOL's performance is enhanced with the addition of equality constraints. Such equality constraints (e.g. pattern nulls) are quite common in antenna weight design practice. In addition, LSSOL does not exploit sparsity of matrices. This is actually an advantage since the mathematical programs that result from translating the design problem, typically, have *full* matrices.

5.2 Description of Awd

Next, we give a brief description of the implemented subset itself. In particular, syntax is similar to the C language. Statements are separated by semi-colons (';'), and groups of

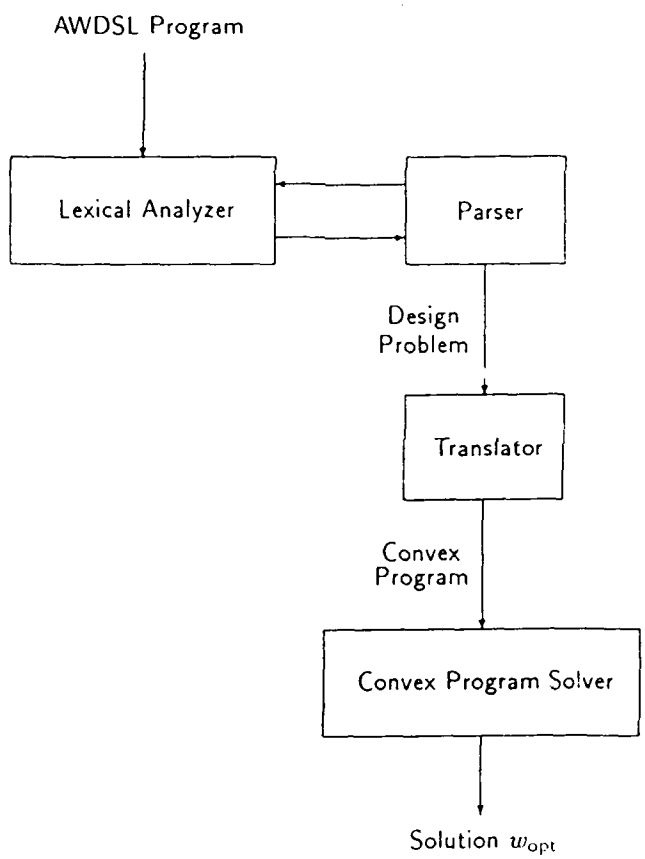


Figure 6: Our implementation of **awd**, a specification language compiler and solver.

statements (multiple statements) are enclosed by braces ('{', '}'). The input is free format. That is, any number of spaces, newlines, and tabs can replace a space.

Looping is allowed and takes the following form,

```
for var = expr1 to expr2 do stmt
for var = expr1 to expr2 step expr3 do stmt
```

Numerical constants are the same as in C. Certain constants such as π (3.14...) and e (2.718...) are defined. Variables (which can take on only scalar floating-point values) and variable names are allowed to assume single lower case letters only. Furthermore, most mathematical functions (cos, sin, *etc.*) are defined.

Next, we shall discuss how each of the three major sections of an AWDSL, array description, constraint, and objective sections, was tailored to fit our implementation of **awd**. For a sample **awd** program, the reader is advised to refer to Example 6.1.

5.2.1 The *array description* section

This section contains the basic information about the array structure and element behavior that allows us to compute any permissible functional. In view of the basic assumptions underlying **awd**, and capabilities of the implemented language, the following information is enough.

- The number of sensor elements in the array.
- The upper and lower bounds on the normalized operational frequency band.
- The x and y locations (in rectangular coordinates) of the array elements in units of nominal wavelength, λ_{nominal}.
- The noise power of each element.

Optionally, an initial estimate of the solution (the weights) may be given in this section. The default starting weights are zeroes.

5.2.2 The *objective* section

The objective function is formed by summing all objective terms appearing in the *objective* section. An objective term is of the form:

```
expr * g;
```

The optional (non-negative) scalar expression *expr* allows relative weighting of different functionals. As was discussed section 5.1.2, an objective functional, g, is either the total noise power in the array output, referred to as `noise_pwr` in **awd**, or the maximum value that the magnitude of the pattern function attains over a certain region, referred to as `max_mag_S`.

5.2.3 The *constraints* section

This section is used to specify design constraints. As was mentioned in section 5.1.2, a design constraint is either a null (`null`), or constraint functional with certain bounds on it. Constraint functionals can be the real part, imaginary part, or the magnitude (`re`, `im`, or `mag`) of an element weight (`W`) or the pattern function (`S`), or may be the maximum value that the magnitude of the pattern function attains over a certain region (`max_mag_S`).

Functional inequalities are allowed to take the forms

g <= *expr* ;
g >= *expr* ;
g == *expr* ;
expr1 <= *g* <= *expr2* ;
abs(*g*) <= *expr* ;

where g is one of the above functionals (other than those involving `mag`, `null`, and `max_mag_S`), and *expr*, *expr1* and *expr2* represent any scalar expressions.

The last form

abs(*g*) <= *expr* ;

is equivalent to

-*expr* <= *g* <= *expr* ;

The special functionals involving `mag`, and `max_mag_S`, can only be used in the form

g <= *expr* ;

In section 5.1.3, we discussed how these constraints are approximated by linear ones.

5.2.4 Producing and reporting awd results

Once the design problem has been successfully compiled into a mathematical program, the mathematical program is solved using LSSOL [GHM*86]. LSSOL usually ends in one of the following results that are reported by **awd**,

- `found strong minimum`: the unique feasible optimal weight values (w) were found.
- `found weak minimum`: non-unique feasible weight values (w) that minimize the objective function were found.
- `solution appears to be unbounded`: happens usually if the weight values (w) in the AWDSL program do not have explicit bounds. It is generally a symptom of an ill-formulated design problem.
- `no feasible point found`: there are no weight values (w) that satisfy the constraints. Some (or all) of the constraints in the *constraints* section must be relaxed.

At the conclusion, the array description, along with the final designed weights, the state of all constraints and the final objective function value are produced and saved in a certain file. Design constraints are printed in that file in the same order in which they were listed in the source file.

The listing for design constraints consists of six-column lines of the form

```
constraint              value    lo_bnd   up_bnd    lambda    status.
```

These columns are as follows.

Column 1: description of what the functional is, such as its name, *e.g.* `max_mag_S`.

Column 2: the value of the functional.

Column 3: the value of the lower bound on the functional.

Column 4: the value of the upper bound on the functional.

Column 5: the value of the Lagrange multiplier for the functional.

Column 6: This field is empty if the functional is within its bounds, or the upper and lower bounds are equal (a nonviolated equality constraint). Otherwise, it will contain one of:

- `lb`
- `ub`
- `violates lb`
- `violates ub`

according to whether the functional is at its lower or upper bound, or exceeds its lower or upper bound, respectively.

Example 6.1 contains a partial listing taken from an actual **awd** run.

6 Examples

6.1 A Linear Array of 20–Elements

In this section it is desired to find a weight vector that will satisfy the constraints exhibited in Figure–7 for a linear array of 20 sensors equally spaced along the y–axis with 0.5λ spacing. The objective is to minimize the total noise power in the received signal. Element power is assumed to be unity for each element.

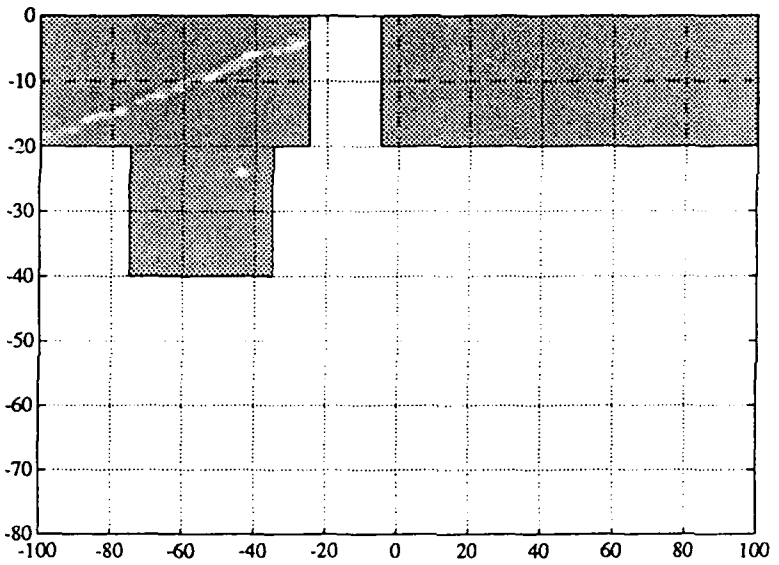


Figure 7: Specifications for array pattern function design.

A possible AWDSL source file for such a problem looks as follows

```
/*
A  linear array of 20 elements spaced at 0.5 wavelength apart along the y-axis.
*/

/* basic definitions: */
#define TARGET          -15
#define NULL            -55
#define BEAM_WIDTH       10
#define NULL_WIDTH       20
#define NULL_GAIN       -40
```

```
#define SIDE_GAIN        -20

array_description{

        n_elements = 20; /* the number of antenna elements in the array */

        for i = 0 to n_elements-1 do {
                element_loc_y(i) = i*0.5 -4.75;
                element_loc_x(i) = 0.0;
                element_pwr(i) = 1.0;
        }
}

minimize{
        noise_pwr;
}

subject_to
{
/* target angle TARGET as defined above */
        re(S(TARGET)) >= 1;

/* a null centered at NULL degrees and of width NULL_WIDTH */
        max_mag_S(NULL-NULL_WIDTH, NULL+NULL_WIDTH, 100)<= inv_dB(NULL_GAIN);

/* no sidelobe has gain higher than SIDE_GAIN dB */
        max_mag_S(-90, TARGET-BEAM_WIDTH) <= inv_dB(SIDE_GAIN);
        max_mag_S(TARGET+BEAM_WIDTH , 90) <= inv_dB(SIDE_GAIN);
}
```

Awd was run on the above source file and a unique optimal weight vector was obtained. The following plots show the resulting **awd**-optimal pattern and how it relates to design constraints.

A partial listing of the **awd** generated output looks as follows

```
                awd  version 2.0

element  x-loc y-loc noise_pwr  complex weight
0        0     -4.75   1       0.02029   -0.0261412
1        0     -4.25   1       0.02168   -0.0306461
2        0     -3.75   1       0.0405     0.00488037
```

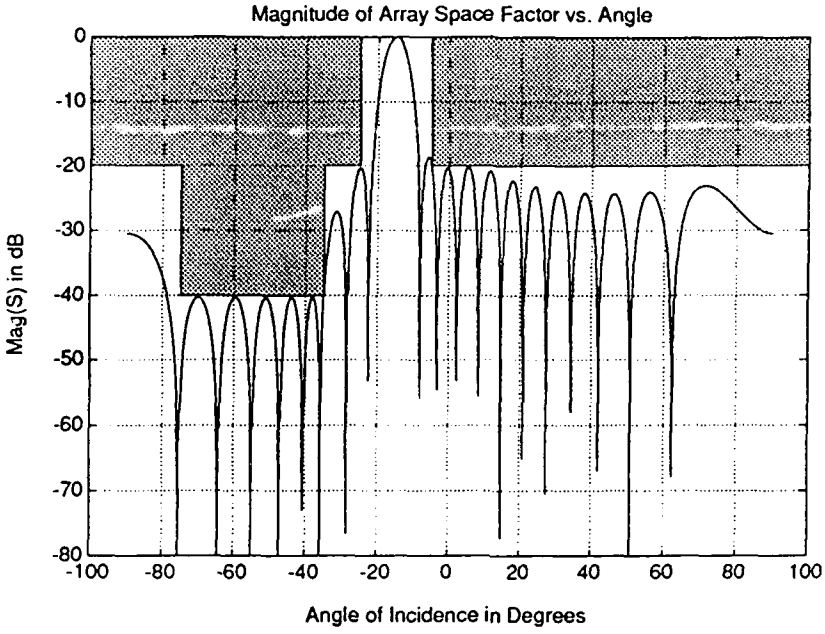


Figure 8: **Awd** optimal design for Example 6.1. Plot of magnitude of pattern function in dB vs. AOI.

```
                              ......
18          0        4.25     1        0.02168     0.0306461
19          0        4.75     1        0.02029     0.0261412

constraint              value lo_bnd up_bnd lambda   status
re(S(-15))                1       1  1e+10    0.1135   lb
                                 ......
max_mag_S(-90,-25,72) 0.09612 0  0.1      -0.01418  ub
       Above constraint attains bounds at:
                          1     -25      0.09612 -0.01418
max_mag_S(-5,90,106)  0.09612 0  0.1      -0.02477  ub
       Above constraint attains bounds at:
                          1     -5       0.09612 -0.01396
                          1     -0.5189 0.09612 -0.008679
                          1      4.858  0.09612 -0.002128
found strong minimum in 998 iterations
objective function value = 0.0538599
total noise power = 0.0538599
```

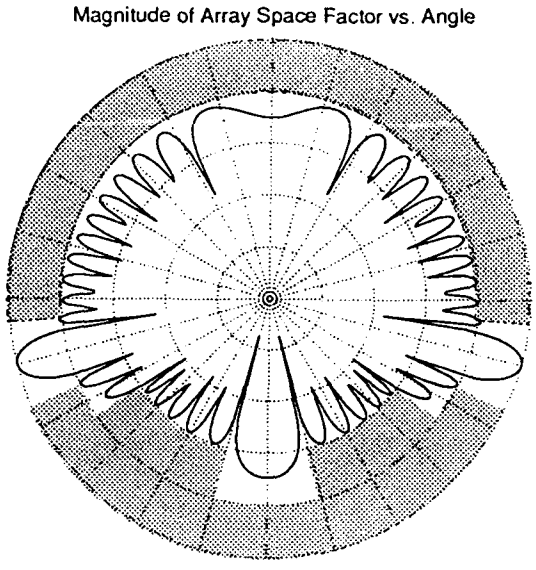


Figure 9: **Awd** optimal design for Example 6.1. Polar plot of magnitude of pattern function in dB.

6.2 A Parabolic Array of 20–Elements

In this section it is desired to find a weight vector that will satisfy the constraints exhibited in Figure–7 for the parabolic array of 20 sensors shown in Figure–10. The objective is still minimizing the total noise power in the received signal.

In addition to the constraints illustrated in the previous example, a constraint on the "back–of–array" gain was included. That is, the magnitude of the pattern function for signals arriving with Angles Of Incidence between 90° and 270° was constrained. This constraint has to do with the asymmetry of the parabolic array as opposed to the linear array. The source file is essentially similar to that for the linear array above, and, thus, will not be listed. It is yet interesting to note that the geometry of the array was specified by placing the statements

```
for i = 0 to n_elements-1 do {
    element_loc_y(i) = i*0.5 - 4.75;
    element_loc_x(i) = (4.75^2-element_loc_y(i)^2)/4.75^2;
}
```

in the *array_description* section, and that the extra "back–of–array" constraints were added by simply adding the line

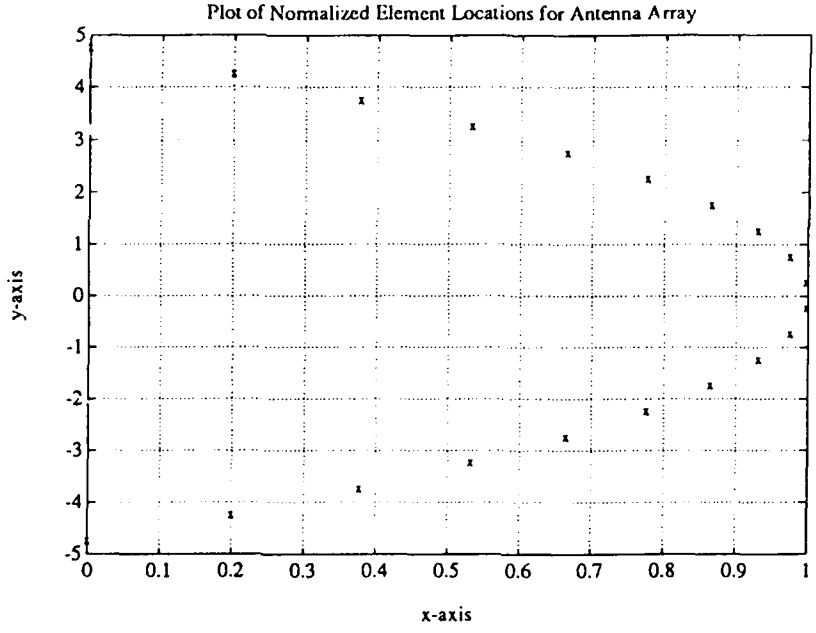


Figure 10: A Parabolic Array of 20–Elements.

```
max_mag_S(90, 270) <= inv_dB(BACK_GAIN);
```

to the *constraints* section of the source file. `BACK_GAIN` was defined to be -10dB.

Running **awd** on the source file for this problem, a unique optimal weight vector was obtained. The following plots show the resulting **awd**–optimal pattern and how it relates to design constraints.

6.3 8–Element Wideband Linear Array

In this section, **awd** is used to design element weights for a linear array of 8 elements responding to narrowband signals with known center frequencies. The elements are equally spaced along the x-axis with a 1.8127 wavelength spacing. Using ***ESPRIT*** [RHR87], it was determined that there were two incident signals. One, considered the target signal, is arriving at an angle of $-9.39°$ relative to broadside, while the other, the interfering signal, is arriving at $1.26°$.

The operational frequency band of interest for the array under consideration is frequencies up to 1.26kHz. If we consider this frequency as the nominal frequency, the target signal will have a normalized frequency of .75. Thus, it was determined that the array pattern should have at least unity gain for signals arriving from the target DOA with the target frequency. Furthermore, it was determined that unwanted signals due to nonlinear effects in the sensors

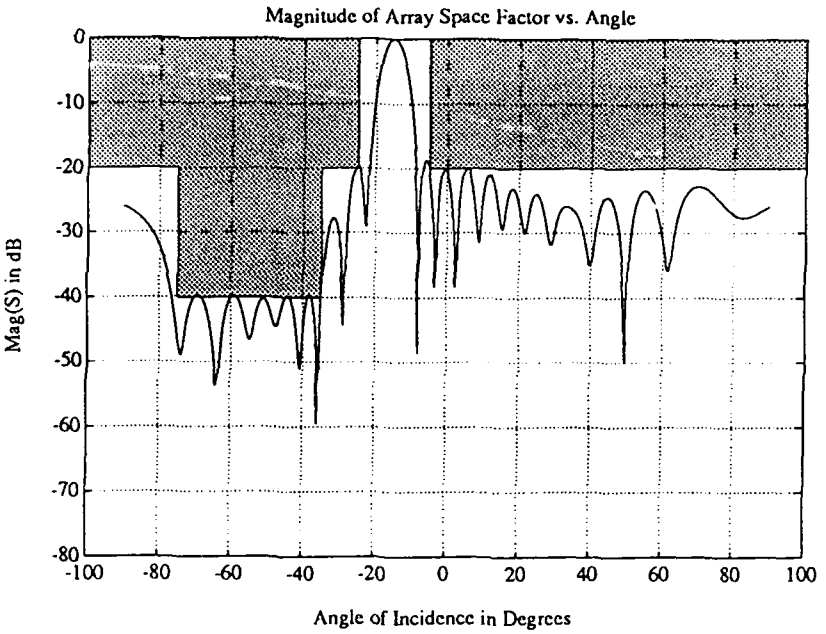


Figure 11: **Awd** optimal design for Example 6.2. Plot of magnitude of pattern function in dB vs. AOI.

were present at frequencies significantly less than the target frequency. Thus, it was required that any signal arriving at the DOA with a normalized frequency of less that 0.5 should encounter at least -20dB suppression.

As for the interfering signal, it was determined that, regardless of its frequency, it should encounter -40dB suppression. In addition, a certain band of possible directions of arrival around the computed interference DOA should also be suppressed. This band is to be 5° wide. That is to say, for any signal arriving at $1.26° \pm 5°$ relative to broadside, with any frequency within the operational frequency band, the array pattern gain should be less than -40dB.

With the above information in mind, an **awd** source program was written, and passed to awd. A comparison between the results obtained by awd and those of ***ESPRIT*** is shown in Figures 13, 14, and 15.

References

[Akg84] M. Akgul. *Topics in Relaxation and Ellipsoidal Methods.* Volume 97 of *Research Notes in Mathematics*, Pitman, 1984.

[BBB*88] S. Boyd, V. Balakrishnan, C. Barratt, N. Khraishi, X. Li, D. Meyer, and S.

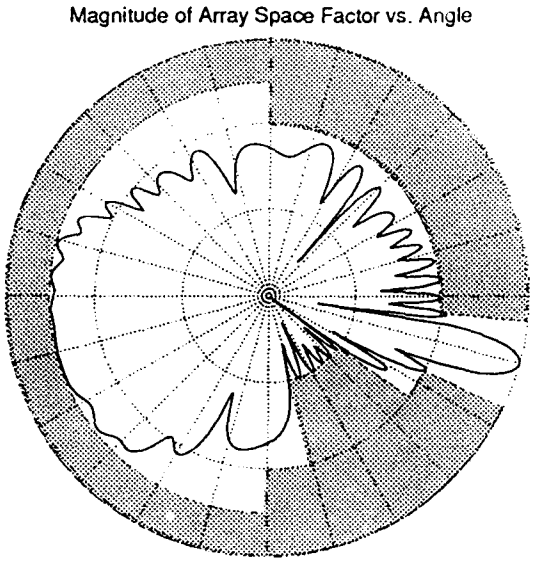


Figure 12: **Awd** optimal design for Example 6.2. Polar plot of magnitude of pattern function in dB.

Norman. A new CAD method and associated architectures for linear controllers. *IEEE Trans. Aut. Control*, AC-33(3):268–283, March 1988.

[BS79] M. Bazaraa and C. Shetty. *Nonlinear Programming, Theory and Algorithms.* John Wiley and Sons, New York, N. Y., 1979.

[BY66] I. Barrowdale and H. Young. Algorithms for best $\mathbf{L}_1$ and $\mathbf{L}_\infty$ linear approximations on a discrete set. *Numerische Mathematik*, 8:295–306, 1966.

[Che71] D. K. Cheng. Optimization techniques for antenna arrays. *Proc. IEEE*, 59(12):1664–1674, December 1971.

[CT65] D. K. Cheng and F. I. Tseng. Gain optimization for arbitrary antenna arrays. *IEEE Trans. Antennas Propag.*, AP-13(6):973–974, November 1965.

[Dol46] C. L. Dolph. A current distribution for broadside arrays which optimizes the relationship between beam width and side-lobe level. *Proc. IRE*, 34:335–348, June 1946.

[EF75] R. J. Evans and T. E. Fortmann. Design of optimal line-source antennas. *IEEE Trans. Antennas Propag.*, AP-23(3):342–347, May 1975.

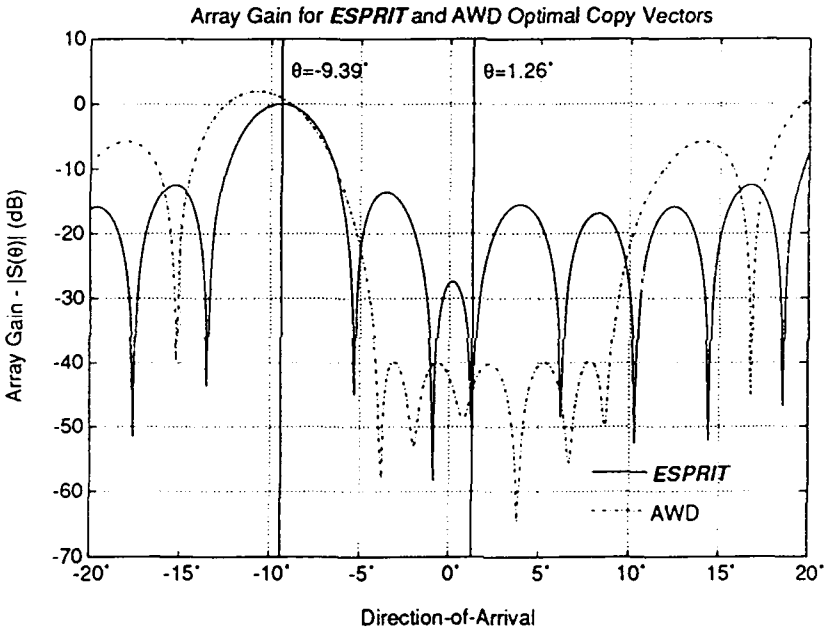


Figure 13: A comparison of combined array pattern obtained using the ***ESPRIT*** optimal weight vector and that obtained using **Awd** for signals arriving with the target frequency of 0.75.

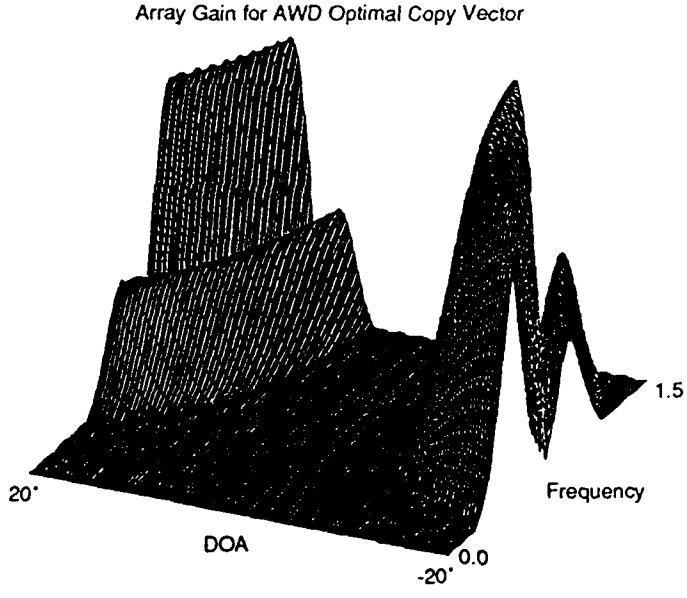


Figure 14: A 3-dimensional plot of the magnitude of the pattern function, using the Awd optimal weight vector, as a function of angle and normalized frequency.

[GHM*86] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for LSSOL (Version 1.0): A FORTRAN Package for Constrained Least-Squares and Convex Quadratic Programming.* Technical Report SOL 86-1, Operations Research Dept., Stanford University, Stanford, California 94305, January 1986.

[GMW81] P. E. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press, London, 1981.

[Joh84] S. C. Johnson. Yacc: Yet another compiler compiler. In *UNIX Programmer's Manual, Supplementary Documents*, University of California, Berkeley, 1984. 4.2 Berkeley Software Distribution.

[Kel60] J. E. Kelley. The cutting-plane method for solving convex programs. *J. Soc. Indust. Appl. Math*, 8(4):703–712, December 1960.

[Khr88] N. M. Khraishi. *AWD User's Manual, Version 2.0.* Prometheus Inc., 103 Mansfield St., Sharon, MA 02067, 1988.

[Khr90] N. M. Khraishi. *A Specification Language Approach to Computer Aided Control System Design of LTI Systems.* PhD thesis, Stanford University, 1990.

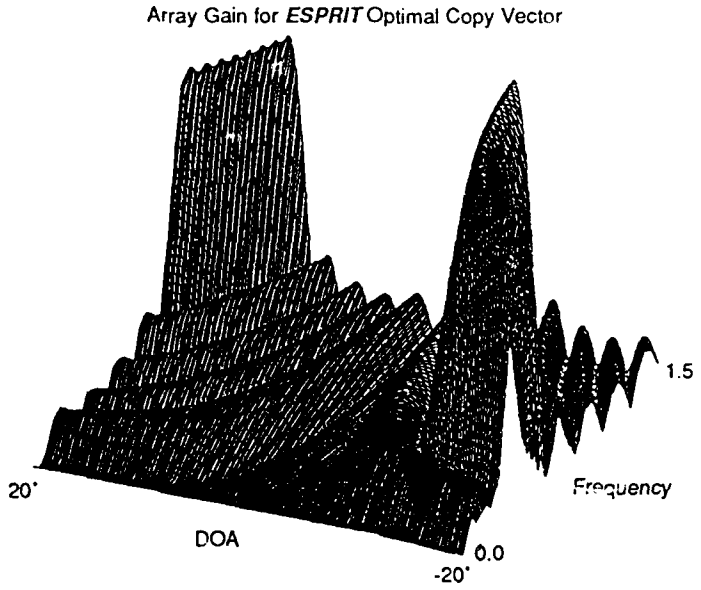


Figure 15: A 3-dimensional plot of the magnitude of the pattern function, using the ***ESPRIT*** optimal weight vector, as a function of angle and normalized frequency.

[KP84] B. W. Kernighan and R. Pike. *The UNIX Programming Environment.* Prentice-Hall, 1984.

[KR78] B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice-Hall, 1978.

[Kra88] J. D. Kraus. *Antennas.* McGraw-Hill, 2 edition, 1988.

[LJS8] R. Leahy and B. Jeffs. *Maximally Sparse Optimization for Array Beamforming and Other Applications.* Technical Report 131, Signal and Image Processing Institute, USC, Powell Hall of Engineering, University Park, Los Angeles, Ca 90089, 1988.

[LS84] M. E. Lesk and E. Shmidt. Lex - A lexical analyzer generator. In *UNIX Programmer's Manual, Supplementary Documents*, University of California, Berkeley, 1984. 4.2 Berkeley Software Distribution.

[LSW66] L. S. Lasdon, D. F. Suchman, and A. D. Waren. Nonlinear programming applied to linear-array design. *J. Acoust. Soc. Am.*, 40(5):1197-1200, 1966.

[Lue69] D. G. Luenberger. *Optimization By Vector Space Methods.* John Wiley and Sons, New York, N. Y., 1969.

[Lue84] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Reading, Mass., 2nd edition, 1984.

[MHM72] G. W. McMahon, B. Hubley, and A. Mohammed. Design of optimum directional arrays using linear programming techniques. *J. Acoust. Soc. Am.*, 51(1):304–309, April 1972.

[Pri53] R. L. Pritchard. Optimum directivity patterns for linear point arrays. *J. Acoust. Soc. Am.*, 25(5):879–891, September 1953.

[RHR87] R. H. Roy. ***ESPRIT*** *– Estimation of Signal Parameters via Rotational Invariance Techniques.* PhD thesis, Stanford University, Stanford, CA., 1987.

[Rib47] H. J. Riblet. Discussion on "a current distribution for broadside arrays which optimizes the relationship between beam width and side-lobe level". *Proc. IRE*, 35:489–492, May 1947.

[Roc70] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, second edition, 1970.

[RWD84] S. Ramo, J. R. Whinnery, and T. Van Duzer. *Fields and Waves in Communication Electronics.* John Wiley and Sons. 2nd edition, 1984.

[SB71] S. M. Sanzgiri and J. K. Butler. Constrained optimization of the performance indices of arbitrary array antennas. *IEEE Trans. Antennas Propag.*, AP-19(4):493–498, July 1971.

[Sch43] S. A. Schelkunoff. A mathematical theory of linear arrays. *Bell Syst. Tech. J.*, 22(1):80–107, January 1943.

[SF85] A. T. Schreiner and H. G. Friedman, Jr. *Introduction to Compiler Construction with UNIX.* Prentice-Hall, 1985.

[Sho85] N. Z. Shor. *Minimization Methods for Non-differentiable Functions. Springer Series in Computational Mathematics*, Springer-Verlag, Berlin, 1985.

[Ste76] B. D. Steinberg. *Principles of Aperture and Array System Design.* John Wiley and Sons, 1976.

[TC67] F. I. Tseng and D. K. Cheng. Gain optimization for arbitrary antenna arrays subject to random fluctuations. *IEEE Trans. Antennas Propag.*, AP-15(3):356–366, May 1967.

[VB72] R. Voges and J. K. Butler. Phase optimization of antenna array gain with constrained amplitude excitation. *IEEE Trans. Antennas Propag.*, AP-20(4):432–436, July 1972.

[Wil76] G. L. Wilson. Computer optimization of transducer-array patterns. *J. Acoust. Soc. Am.*, 59(1):195–203, January 1976.

[Wil78] G. L. Wilson. The design of antenna arrays with tapered sidelobe heights. *IEEE Trans. Antennas Propag.*, AP-26(2):345–347, March 1978.

[WLS67] A. D. Waren, L. S. Lasdon, and D. F. Suchman. Optimization in engineering design. *Proc. IEEE*, 55(11):1885–1896, November 1967.

Prometheus Inc.
Final Report
11 July 1990

Section IX
awd *User's Manual*

AWD User's Manual
Version 3.0

Nasser M. Khraishi[1]
Prometheus Inc.
103 Mansfield St.
Sharon, MA 02067

July 11, 1990

[1]The author is also with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University.

Contents

1 INTRODUCTION

Awd is a *computer aided design* program for designing antenna element complex weights for arbitrary arrays of omnidirectional elements with fixed relative locations and negligible mutual interactions. Elements are assumed to have constant complex gain over a predetermined (finite) frequency band of operation. Currently, signal sources are assumed to be farfield. An array configuration for the case when all elements and signals lie in a plane is shown in Figure 1.

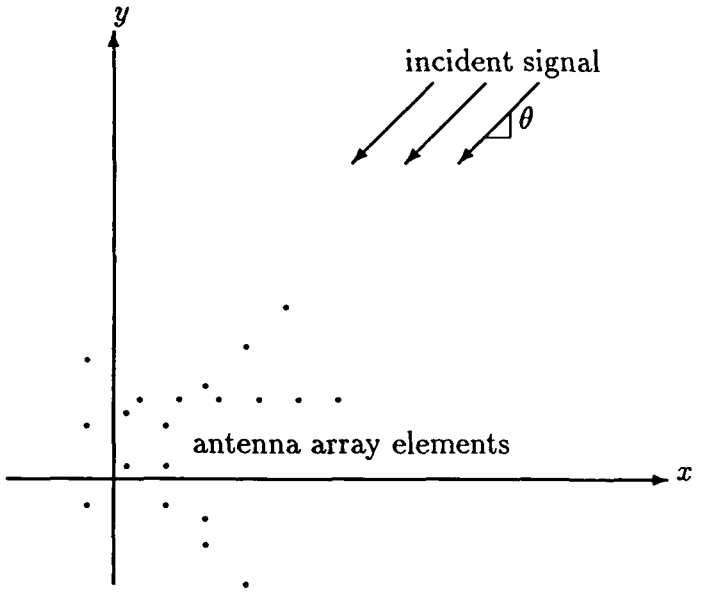


Figure 1: Antenna array under consideration.

Awd takes as its input an antenna weight design problem written in a specialized *antenna weight design specification language.* **Awd** then *compiles* this problem into a convex mathematical program. The resulting mathematical program is *solved* using a convex program solver. The output of **awd** is a set of complex numbers, one per antenna element, that constitutes the *weight vector.* These complex weights determine the relative scaling and phase shift by which the output of each antenna element contributes to the *combined array output.* This combined array output is a scalar that is the inner product of the **awd** designed weight vector and the vector output of the array.

2 DESCRIPTION

Awd is invoked as follows:

`awd [-l]` *filename*

where *filename* is a file that contains a specific design problem specified using an *antenna weight design specification language* that is discussed below. The design problem is then compiled into a mathematical program. This is the *compilation phase.* Note that in the resulting mathematical program, the real and imaginary parts of an element complex weight are considered as two independent real optimization variables.

If no errors occur in the compilation, an optimization routine is called to solve the mathematical program. This is the *optimization phase.* The first step in this phase is to determine the *feasibility* of the mathematical program. If the program is feasible, *optimal* values of the design weights are found. The optimization code used in the current version of **awd** is LSSOL [GHM*86], a library of convex programming routines for linear, quadratic, and least squares optimization.

Upon completion, the antenna array specification, the weights designed, and the values of the constraint functionals and their upper and lower bounds are written into the file *filename*`.out`. The values of the design variables are written into the file *filename*`.w` if they are optimal, or into the file *filename*`.last.w` if the optimization failed.

If the `-l` option is specified, the log information from the convex program solver LSSOL (the optimization code used) is written to the file *filename*`.lslist`. Extra options for LSSOL may be specified in the file *filename*`.lsoptn`.

3 PREPARING TO USE AWD

The following information concerning the design problem is required to set up the input to **awd**:

- a finite operational frequency range,
- pre-steering information,
- and, antenna element locations.

First of all, a finite operational frequency band of interest is selected by setting lower (f_{lower}) and upper (f_{upper}) frequencies. Currently, antenna element responses are assumed to be constant over the selected band. Note that for narrowband array design problems, the lower and upper frequencies can be equal. All frequencies are specified in units of a certain nominal frequency, f_{nominal}. A good choice of f_{nominal} is the center frequency of the antenna element operational frequency band. A frequency of interest (f) in physical units (Hertz) is input to **awd** as a normalized frequency, $\bar{f} = f/f_{\text{nominal}}$. In other words, a frequency of 0.5 specifies a frequency of half the nominal frequency.

If the array is *pre-steered* to a certain reference direction (say, azimuthal angle θ_{ref}, and elevational angle ϕ_{ref}) for signals of a certain normalized frequency (say f_{ref}), this information should be made available to **awd**.

Finally, locations (in rectangular coordinates) of the array elements must be given. The positions are to be specified in units of λ_{nominal}, the wavelength corresponding to the nominal operation frequency (f_{nominal}) and the underlying speed of propagation (c) in the medium ($\lambda_{\text{nominal}} = c/f_{\text{nominal}}$).

It is assumed that the designer is concerned with the *array space factor*[1] [RWD84, Sch43]. This is a function of azimuthal angle, θ, elevation angle, ϕ, (normalized) signal frequency, f, [2] complex weights, W_i's, and (normalized) element location vectors, $P_i = [x_i\ y_i\ z_i]^T$. The function is given by:

$$S(\theta, \phi, f) = \sum_{i=0}^{n-1} W_i \exp j(k - k_{\text{ref}})^T P_i, \tag{1}$$

where n is the number of elements in the array,

$$k_{\text{ref}} = 2\pi f_{\text{ref}} \begin{bmatrix} \sin\phi_{ref}\cos\theta_{ref} \\ \sin\phi_{ref}\sin\theta_{ref} \\ \cos\phi_{ref} \end{bmatrix} \tag{2}$$

and

$$k = 2\pi f \begin{bmatrix} \sin\phi\cos\theta \\ \sin\phi\sin\theta \\ \cos\phi \end{bmatrix}. \tag{3}$$

If no pre-steering information are given, k_{ref} will be assumed to be a zero vector (which is equivalent to assuming a zero reference frequency).

The user must also specify the total noise power output p_i of each element in the array over the operational frequency band. Each p_i can be thought of as the integral of the noise power density of the i^{th} element over the operational frequency band. This noise can be due to thermal, amplifier, and/or spatial noise. The total noise power in the combined array output is given by:

$$N = \sum_{i=0}^{n-1} p_i |W_i|^2. \tag{4}$$

This assumes that noises in the different array elements are *independent.*

Finally, the user must determine design objectives and constraints. In the current version of **awd**, design objectives are allowed to be one (or a weighted sum) of the following:

- $\max\limits_{f_1 \le f \le f_2} \max\limits_{\phi_1 \le \phi \le \phi_2} \max\limits_{\theta_1 \le \theta \le \theta_2} |S(\theta, \phi, f)|$, maximum value that the magnitude of the array space factor assumes between azimuth angles θ_1 and θ_2, for any elevation angle between ϕ_1 and ϕ_2, and any (normalized) frequency between f_1 and f_2.

[1] The array space factor is sometimes referred to as the *array directivity pattern.*

[2] From now on it is assumed that all frequencies are normalized ($\bar{f} = f/f_{\text{nominal}}$) frequencies unless otherwise stated.

- total noise power in the combined array output (*cf.* equation (4)).

Design constraints are constructed from the following expressions:

- $\max_{f_1 \le f \le f_2} \max_{\phi_1 \le \phi \le \phi_2} \max_{\theta_1 \le \theta \le \theta_2} |S(\theta, \phi, f)|$, same as above.
- real, imaginary, or magnitude of $\alpha S(\theta, \phi, f)$, where α is some complex weighting factor, θ is some azimuthal angle, ϕ is an elevation angle, and f is a (normalized) frequency.
- real, imaginary, or magnitude of αW_i, where α is some complex weighting factor, and W_i is the complex weight factor for element i.

All the expressions above can be shown to be *convex* functionals of the real and imaginary parts of the weight factors W_i's.

Another design constraint can be imposed by setting $S(\theta, \phi, f)$ to zero at given incident angles, θ and ϕ, for some (normalized) frequency, f. This is termed *placing a null* at θ, ϕ, and f. It can be shown that placing a null at angles, θ and ϕ, and frequency, f, is an *affine* constraint on the real and imaginary parts of the weight factors W_i's.

4 THE ANTENNA WEIGHT DESIGN SPECIFICATION LANGUAGE (AWDSL)

The Antenna Weight Design Specification Language (AWDSL) that the program **awd** compiles and runs consists of three major sections, *array description*, *objective function*, and *design constraints*. Each of these sections consists, as will be detailed below, of a key word, followed by a left brace '{', followed by a group of statements (some of which is characteristic to the certain section), and ended by a right brace '}'.

A typical AWDSL program will look as follows:

```
array_description {
        /* description of array geometry and element noise powers*/
        ...
}
minimize {
        /* description of design objective */
        ...
}
subject_to {
        /* description of design constraints */
        ...
}
```

Each of these sections is described below in full detail.

4.1 Statement Syntax

Syntax is similar to the C language: statements are separated by semi-colons (';'), and groups of statements are enclosed by braces ('{', '}'). The input is free format: any number of spaces, newlines, and tabs can replace a space. **Awd** passes the source file *filename* (which contains the AWDSL program) through the C preprocessor *cpp*, so all the standard preprocessor functions are available, for example `#define` and `#include`. C-style comments (enclosed between "`/*`" and "`*/`") are permitted. As in C, comments do not nest.

A statement can be any of the following.

- **Multiple statements:** as in C, a statement can be replaced by any number of statements enclosed in braces ('{', '}').
- **Assignment:** real variables can be assigned the values of expressions. See below.
- **For loop:** looping takes the following forms:

```
for var = expr1 to expr2 do stmt
for var = expr1 to expr2 step expr3 do stmt
     /* the word 'do' is synonymous to 'sum,' thus
     the above can be written as:  */
for var = expr1 to expr2 sum stmt
for var = expr1 to expr2 step expr3 sum stmt
```

The variable *var* (a lower case letter) is assigned the value of expression *expr1*. *Expr2* sets the final value of *var*. *Expr3* sets the increment of *var* each time around the loop. The default step size is 1.0. The expressions *expr2* and *expr3* are only evaluated when the for loop starts. Thus, variable step sizes are not possible. For example, the following loops execute the statement *stmt* 10 times:

```
for i = 0 to 9 do stmt
for i = 0 to 0.9 step 0.1 do stmt
```

- **Array-description statement:** is used to describe the array geometry and characteristics. Array-description statements can only appear in the *array description* section. An array-description statement can be one of the following:

```
n_elements = expr;
lower_freq = expr1;
upper_freq = expr1;
element_loc_x(expr2) = expr3;
element_loc_y(expr2) = expr3;
element_loc_z(expr2) = expr3;
element_pwr(expr2) = expr3;
```

```
element_w(expr2) = (expr3, expr4);
reference = (expr5,expr6,expr7);
mag_2_lin = expr8;
```

where *expr* is the number of antenna elements in the array, *expr1* is any non-negative valued expression, *expr2* represents a given element index and must evaluate to a value between 0 and `n_elements`-1, and *expr3* and *expr4* are any real valued expressions. *Expr5*, *expr6*, and *expr7* give the azimuth, elevation and frequency information of the reference signal for pre-steered arrays. *Expr8* assumes the value of 2 or 4 depending on whether it is wished to approximate a circle by a square or an octagon.

- **Functional statement:** can be either a *constraint statement*, which is used to specify a design constraint in the *constraints* section of the source file, or an *objective term*, where the optimization objective is the sum of the objective terms that are specified in the *objective* section of the source file. These are explained below.

4.2 Constants, Variables and Expressions

Numerical constants are the same as in C. The mantissa can have an optional fraction part. An exponent can be specified as '`e`' or '`E`', followed by an optional '`+`' or '`-`', followed by an integer. There can be no spaces, tabs, or new-lines in a constant. "`%pi`" and "`%e`" are replaced by π (3.14...) and e (2.718...) respectively. Also, "`infinity`" is replaced by (1.0e10). In addition, since angles in the program are assumed to be in degrees, two constants are made available to the user. These two constants are "`deg_2_rad`" and "`rad_2_deg`," which are replaced by $(\pi/180)$ and $(180/\pi)$ respectively.

One more constant, called "`correction_factor`," can only be used in the *objective* and *constraints* sections, but not in the *array description* section. This constant, as will be discussed below, has to do with the fact that **awd** approximates magnitude constraints, which are quadratic in nature, by a number linear constraints. The constant "`correction_factor`" is internally set to $1/\sqrt{\cos(22.5)}$ if the magnitude constraint is to be approximated by 4 pairs of linear constraints, and to $1/\sqrt{\cos(45)}$ if the magnitude constraint is to be approximated by 2 pairs of linear constraints.

Variables (which can take on only scalar floating-point values) are restricted to names with one lower case letter. The variables can be assigned real numbers (doubles) and used in scalar expressions. For example,

```
x = 1 + 2 * 5.2;
y = x + 3;
```

will assign 11.4 to `x` and 14.4 to `y`.

The binary operators '`+`', '`-`', '`*`', '`/`', '`^`' (plus, minus, multiply, divide and exponentiation) operate between two scalar expressions. They associate left to right and have the usual precedence ('`^`' highest, followed by '`*`', '`/`' (equal), followed by '`+`', '`-`' (equal)). Parentheses

can be used in the usual manner to force order of evaluation. For example, all of the following statements assign 8.0 to the variable a.

```
a = 2 * 2 + 2 * 2;
a = 2 * (1 + 3);
a = 0.5 ^ -3;
a = 2 ^ 3;
a = 4 / 2 * 4;
```

The functions **sin**, **cos** and **tan** return the corresponding trigonometric function values of *degree* arguments. The functions **asin**, **acos** and **atan** return the inverse trigonometric function values (as in the C mathematical library), except that the angles are in degrees. If **atan** is called with two arguments (comma (',') separated), it will be equivalent to the C library function **atan2** (with the result converted to degrees, resulting in an answer between -180 and 180 degrees). The functions **exp**, **log**, **log10**, **dB**, **inv_dB**, **sqrt** and **abs** return the exponential and logarithm (base e), the logarithm (base 10), the decibel ($20\log_{10}$) and its inverse, positive square root and absolute values of their arguments (respectively). In addition, the functions **min** and **max** take two arguments (comma (',') separated) and return the smaller or larger argument respectively. All of these functions take expressions which evaluate to real scalars as arguments. Finally, **floor** finds the greatest integer less than the given argument. For example, all of the following statements assign a number close to 2.0 to the variable b.

```
b = sqrt(4);
b = log10(100);
b = 2 * min(5 * cos(1e-10), exp(max(0*x,-1)));
b = floor(2.5);
w = rad_2_deg * %pi;
b = 2 * (cos(w/4) * cos(w/4) + sin(w/4) * sin(w/4));
```

Expressions in **awd** can be used in place of a constant anywhere in the source file *filename*. Thus, for example, expressions can be used for array indices (the compiler will round indices to the nearest integer).

4.3 The *Array Description* Section

This section must appear first in the program. As the name suggests, this section contains information about the array structure. The number of elements (**n_elements**) in the array is the first thing to be specified. Optionally, the upper (**upper_freq**) and lower (**lower_freq**) bounds on the operational frequency band are specified next. If either **upper_freq** or **lower_freq** is not specified, it will be defaulted to unity (the nominal frequency). Thus for narrowband elements, the user can ommit the specification of such bounds and they will be defaulted to the nominal frequency. The array geometry is then given by specifying the x, y, and z locations (in rectangular coordinates) of the array elements in units of nominal

wavelength, λ_{nominal}. Finally, the noise power of each element, p_i is specified. Optionally, the initial element weights can be specified in this section. The above information is set using *array-description statements* of the form discussed above.

If the array is pre-steered to a certain direction for a given (scaled) signal frequency, such (optional) information should also be made available. If no pre-steering information are given, k_{ref} will be assumed to be a zero vector. That is, an infinite reference frequency will be assumed. An *array description* section thus has the form:

```
array_description{
     /* specify the number of antenna elements in the array */
     n_elements = number of elements;

     /* specify the pre-steering information */
     reference = (reference azimuth,reference elevation, reference frequency);

     /* how to approximate magnitude constraints*/
     mag_2_lin = 2 for a square or 4 for an octagon;

     /* specify the lower and upper bounds on operational frequency band */
     lower_freq = lower bound;
     upper_freq = upper bound;

     /* specify the x locations */
     element_x_loc(0) = x-loc of element 0;
          .......
     element_x_loc(n_elements - 1) = x-loc of element n_elements-1;

     /* specify the y locations */
     element_y_loc(0) = y-loc of element 0;
          .......
     element_y_loc(n_elements - 1) = y-loc of element n_elements-1;

     /* specify the z locations */
     element_z_loc(0) = z-loc of element 0;
          .......
     element_z_loc(n_elements - 1) = z-loc of element n_elements-1;

     /* specify the element noise powers */
     element_pwr(0) = power of element 0;
          .......
     element_pwr(n_elements - 1) = power of element n_elements-1;

     /* specify the initial weights */
```

```
    element_w(0) = (real part , imaginary part);
        .......
    element_w(n_elements - 1) = (real part , imaginary part);
}
```

Note that the array elements are indexed from 0 to `n_elements`−1.

The `mag_2_lin` (optional) value specifies to the program the method of approximating a magnitude constraint. A value of 2 means that the magnitude constraint (which resembles a circle) is to be approximated by two pairs of linear constraints (a square), while a value of 4, the default, indicates that the magnitude constraints should be approximated by four pairs of linear constraints (an octagon).

The `element_w` statements above allow an initial estimate of the solution to be given. If **awd** is being re-run on a source file which has had a small number of constraints changed, it is likely that the old optimal weight values will be a good starting point for the *optimization phase.* This will speed up the *optimization phase.* If `element_w` is not specified, a zero starting point will be used.

The user is advised to store the above coefficients in separate files that can be conveniently included in the file *filename* using the "`#include`" facility of the C preprocessor. For example, if the coefficients for x-locations are in the file "`problem.x`," then the following should appear in the source file:

```
    #include "problem.x"
```

It should be noted that the C preprocessor (cpp) requires that the symbol '`#`' is the first character on the line.

The following is a typical *array description* section, which describes a circular array of ten narrowband elements:

```
array_description {
/* the number of elements is the first thing to declare */
        n_elements = 10;

/* specify a circular array of radius equal to half wavelength */
        for i = 0 to n_elements -1 do {
                element_x_loc(i)  = 0.5*cos(360/n_elements * i);
                element_y_loc(i)  = 0.5*sin(360/n_elements * i);
                element_z_loc(i)  = 0;
        }

/* the other information is included in separate files */
#include problem.p
#include problem.w
}
```

The file `problem.p` contains the noise power of the array elements, and the file `problem.w` contains initial starting weights. Note that, as in C, to loop over two or more statements, the statements must be enclosed by braces.

4.4 The *Objective* Section

This is the second section of the AWDSL program and is optional. If the *objective* section is missing, **awd** will understand that the problem is a *feasibility* problem and simply try to satisfy the constraints in the *constraints* section (discussed below), provided the latter section exists. The objective function is formed by summing all the objective terms appearing in the *objective* section.

An objective term is of the form:

expr * *objective_functional*;

The optional scalar expression *expr* allows relative weighting of the different functionals. The *objective_functional* must be one of the following:

- `noise_pwr`: total noise power in the combined array output (as discussed above).
- `max_mag_S`(*θ-range*)(*φ-range*)(*f-range*): the maximum value that the magnitude of the space factor assumes on certain points within the given azimuthal (θ), elevation (ϕ), and frequency (f) ranges. If one of the ranges is omitted, it is replaced by a single default value. The default value for the azimuth is 0°, the value for the elevation is 90° (parallel to the x-y plane), and the default value for the frequency range is 1 (corresponding to the nominal frequency).

  Each range consists of a lower bound, upper bound, and an integer resembling the number of points at which the given range is to be sampled. The number of intermediate sample points is optional. There are no default values for the lower and upper bounds for any of the above ranges, but the value for the number of sample points is optional and is defaulted to 100 points for the azimuth range, 50 for the elevation range, and 10 for the frequency range. Such values are arbitrary and thus the user should avoid using such defaults.

The following is a sample *objective* section which explains the use of the above functionals:

```
minimize {
/* total output noise */
        noise_pwr;

/* magnitude of space factor on a given azimuth angle range */
        max_mag_S(15, 345);

/* the above is equivalent to
```

```
        max_mag_S(15, 345,100)()();
or
        max_mag_S(15, 345,100)(90)(1);
*/

/* magnitude of space factor for given azimuthal and elevational ranges */
        max_mag_S(-75, -30, 40)(40, 140, 40);

/* the above is equivalent to
        max_mag_S(-75, -30, 40)(40, 140, 40)();
or
        max_mag_S(-75, -30, 40)(40, 140, 40)(1);
*/

/* magnitude of space factor for given azimuthal and frequency ranges */
        max_mag_S(75, 105, 40)()(0.8, 1.2, 20);

/* the above is equivalent to
        max_mag_S(75, 105, 40)(90)(0.8, 1.2, 20);
*/
}
```

4.5 The *Constraints* Section

This section is used to specify design constraints. This is the third and final part of the program. If no such section exists, the problem will be an *unconstrained optimization* problem.

The following functionals may be referred to in the *constraints* section:

- `re(`*scl_fctr*`*W(`*i*`))`, `im(`*scl_fctr*`*W(`*i*`))`, `mag(`*scl_fctr*`*W(`*i*`))`: the real, imaginary, and magnitude of the complex weight of element i.

- `null(`θ`, `ϕ`, `f`)`: the space factor has a null at azimuth angle θ, elevation angle ϕ, and (normalized) frequency f, i.e. $S(\theta, \phi, f) = 0$. If f is omitted, it will be assumed to have unity value (the nominal frequency). If ϕ is omitted, it will be defaulted to 90°, that is a plane parallel ot the x-y one. The default for θ is 0°.

- `re(`*scl_fctr*`*S(`θ`, `ϕ`, `f`))`, `im(`*scl_fctr*`*S(`θ`, `ϕ`, `f`))`, `mag(`*scl_fctr*`*S(`θ`, `ϕ`, `f`))`: the real, imaginary, and magnitude of the space factor with respect to incident azimuth angle θ, elevation angle ϕ, and (normalized) frequency f. If f is omitted, it will be assumed to have unity value (the nominal frequency). If ϕ is omitted, it will be defaulted to 90°, that is a plane parallel ot the x-y one. The default for θ is 0°.

- `max_mag_S(`θ*-range*`)(`ϕ*-range*`)(`f*-range*`)`: same as the functional discussed in the *objective* section.

In the above, *scl_fctr* is a complex (optional) scale factor that has the form (*real part, imaginary part*). If no scale factor is given, it is assumed to be (1.0, 0.0), i.e 1.

Functional inequalities are allowed to take the forms

g <= *expr* ;
expr >= *g* ;
g >= *expr* ;
expr <= *g* ;
g == *expr* ;
expr == *g* ;
expr1 <= *g* <= *expr2* ;
expr1 >= *g* >= *expr2* ;
abs(*g*) <= *expr* ;
expr >= abs(*g*);

where *g* is one of the above functionals (other than those involving mag, null, and max_mag_S), and *expr*, *expr1* and *expr2* represent any scalar expressions.

The last two forms

abs(*g*) <= *expr* ;

and

expr >= abs(*g*) ;

are equivalent to

-*expr* <= *g* <= *expr* ;

The special functionals involving mag, and max_mag_S, can only be used in the form

g <= *expr* ;

where *expr* is a non-negative valued expression.

A mag functional inequality is replaced by the four inequalities

-*expr*/correction_factor <= re <= *expr*/correction_factor;
-*expr*/correction_factor <= im <= *expr*/correction_factor;
-*expr*/correction_factor <= (re + im)/sqrt(2) <= *expr*/correction_factor;
-*expr*/correction_factor <= (re - im)/sqrt(2) <= *expr*/correction_factor;

where "correction_factor" is a constant that is set to $1/\sqrt{\cos(22.5)}$, and the above approximation yields a maximum of ± 0.34 dB error if the mag_2_lin parameter is set to 4 (the default).[3]

A sample *constraints* section is given below:

[3]If "correction_factor" is set to $1/\sqrt{\cos(45)}$ if mag_2_lin is set to 2, and the program uses only the first two inequalities in approximating a magnitude constraint.

```
subject_to {
/* bounds on W coefficients */
        for k=0 to n_elements-1 do
                mag(W(k)) <= 1.0;

/* a null at 90 degrees incident angle and nominal frequency */
        null(90);

/* the above null can be rewritten as */
        re(S(90)) == 0.0;
        im(S(90)) == 0.0;

/* or it can be rewritten as */
        mag(S(90)) <= 0.0;

/* constrain magnitude of space factor on given azimuth and frequency ranges */
        max_mag_S(60, 300, 100)(90)(0.8, 1.2, 5) <= 0.1;

/* require that the array space factor has a magnitude higher
       than one in the zero direction for a range of frequencies */
for f = 0.7 to 1.3 step 0.05 do
         re(S(0,90,f)) >= 1.0;
}
```

5 OPTIMIZATION

Once the design problem has been compiled into a mathematical program, the *compilation phase* is complete and the *optimization phase* can be executed. This is where the mathematical program is solved. Using LSSOL [GHM*86], a convex quadratic program solver from the Systems Optimization Laboratory of the Department of Operations Research at Stanford University, the objective function is minimized subject to the affine constraints given.

The *optimization phase* usually ends in one of the following results:

- `found strong minimum`: the unique feasible optimal weight values (W) were found.
- `found weak minimum`: non-unique feasible weight values (W) that minimize the objective function were found.
- `solution appears to be unbounded`: happens only if the weight values (W) in the AWDSL program do not have explicit bounds. It is generally a symptom of an ill-formulated design problem.
- `no feasible point found`: there are no weight values (W) that satisfy the constraints. Some (or all) of the constraints in the *constraints* section must be relaxed.

With the first two outcomes, the optimal weight values (W), are written into the file *filename*.`w`. This file is written in the same format as required in the *array description* section. Thus, W values can be conveniently used as the starting point for the optimization in a subsequent run of **awd** by placing the line

`#include "`*filename*`.w"`

in the *array description* section. This will make the *optimization phase* of a subsequent run of **awd** run faster (if the constraints or objectives have not been changed significantly).

With the remaining outcomes of the *optimization phase*, the last W values are written into the file *filename*`.last.w` instead. This allows the values to be inspected without the result of any successful run (in *filename*`.w`) being destroyed.

Regardless of the outcome of the *optimization phase*, the array description, along with the produced weights, the state of all constraints and the final objective function value are written into the file *filename*`.out`. The constraint functionals are printed in that file in the same order in which they were listed in the source file.

As far as the file *filename*`.out` is concerned, the constraint functionals can be divided into two categories; those for which only one line in the file *filename*`.out` is printed, and those with one, or more printout lines (depending on the results of the optimization).

For the first type of constraint functionals, the line is formatted into six (6) columns separated by spaces. A description of these columns follows.

Column 1: one of the following functionals

- `re(`*scl_fctr*`*W(`i`))`,
- `im(`*scl_fctr*`*W(`i`))`,
- `mag(`*scl_fctr*`* W(`i`))`,
- `re(`*scl_fctr*`*S(`θ`, `ϕ`, `f`))`,
- `im(`*scl_fctr*`*S(`θ`, `ϕ`, `f`))`,
- `mag(`*scl_fctr*`*S(`θ`, `ϕ`, `f`))`,
- `null(`θ`, `ϕ`, `f`)`.

If ϕ and/or f are not printed, it is implied that they assume the default values of 90° (the horizontal plane) and unity (the nominal frequency) respectively.

Column 2: the value of the functional.

Column 3: the value of the lower bound on the functional.

Column 4: the value of the upper bound on the functional.

Column 5: the value of the Lagrange multiplier for the functional.

Column 6: This field is empty if the functional is within its bounds, or the upper and lower bounds are equal (an unviolated equality constraint). Otherwise, it will contain one of:

- `lb`
- `ub`
- `violates lb`
- `violates ub`

according to whether the functional is at its lower or upper bound, or exceeds its lower or upper bound, respectively.

For example, the following fragment from the *constraints* section of the source file *filename*

```
re(S(0)) >= 1.0;
mag(S(0)) <= 1.5;
null(270,90,0.9);
```

may produce the following lines in *filename*`.out`

```
re(S(0))                    1       1       1e+10       1.04       lb
mag(S(0))                   1.082   0       1.5         0
null(270,90,0.9)            0       0       0           0
```

The second type of constraint functionals includes the functional `max_mag_S`(*θ-range*)(*φ-range*)(*f-range*). For this functional, the first line is formatted exactly as in the case above. If the bounds are met or violated, more lines will be printed. The first extra line would be either,

```
Above const. int attains bounds at:
```

or,

```
Above constraint violates bounds at:
```

as appropriate. The lines that follow would be five-column-printouts with:

Column 1: the azimuthal angle at which the functional attained or violated its constraints.

Column 2: the elevational angle at which the functional attained or violated its constraints.

Column 3: the (normalized) frequency at which the functional attained or violated its constraints.

Column 4: the value of the functional at that point.

Column 5: the Lagrange multiplier of the functional at that point.

For example, the following fragment from the *constraints* section of the source file *filename*

```
max_mag_S(90,270,50)(90)(1.1) <= 0.1;
```

may produce the following lines in *filename*.out

```
max_mag_S(90,270,50)              0.1035  0       0.1       1.87      ub
                        Above constraint attains bounds at:
                         90    90   1.1     0.1035     1.24
                        180    90   1.1     0.0999     0.63
```

It is only fair to warn the user that, due to the nature of optimization algorithms in general, it may be the case that the algorithm did not detect all of the constraints at which the bounds are attained. Thus, in the above example, it may be the case that the bounds are also attained at other angles not shown above. However, all violations are detected.

6 FILE SUMMARY

Input files

filename	the source file name given as argument to **awd**.
filename.`lsoptn`	extra LSSOL options (optional).

Output files

filename.`w`	W values after successful optimization.
filename.`last.w`	W values after unsuccessful optimization.
filename.`out`	listing file for variable and constraint values.
filename.`lslist`	LSSOL listing if `-l` option given.

7 HINTS

7.1 LSSOL Options

There are many optional parameters that the user can specify in the file *filename*.`lsoptn`, [GHM*86]. In particular, `crash tolerance` and `feasibility tolerance` have significant effects on the number of iterations that LSSOL takes to find the optimal solution. Thus, it may be unwise to let the *optimization phase* of the program run for long periods of time, since there may be an option that can be set in the `lsoptn` file to make the program run faster on your specific problem. See the LSSOL manual for all the allowable options and their effects on the behavior of the code.

Options in the *filename*.`lsoptn` file can be set with statements of the form:

```
# this is a sample lsoptn file
crash tolerance = 1.0e-4
```

```
feasibility tolerance = 1.0e-6
feasibility phase iteration limit = 100
optimality phase iteration limit = 100
print level = 5
```

Lines starting with the character '#' are treated as comments and are ignored.

7.2 Problem Size

The user should note that **awd** may generate a large number of linear constraints. Thus, it is advisable to be economical with the size of problems handled using the code. This limitation is due to LSSOL (see [GHM*86]).

7.3 Speed

Most of the number crunching is done in the optimization code LSSOL. LSSOL uses a standard set of "BLAS" (Basic Linear Algebra Subroutines). Those can be freely replaced by any appropriate BLAS fine tuned to the user's machine. In particular, by profiling the code, we found out that most of the time is spent in two specific BLAS, "DAXPY" and "DDOT." Running **awd** on the "narrowband" example shown below, it was found that 99.4% of the time was spent in solving the mathematical program, 39% of the total time was spent in "DAXPY," and 20% of the time was spent in "DDOT."

7.4 Assuring Sensible Answers

Although **awd** will generate correct results for any problem of reasonable size, the designed weights need not be sensible from an engineering point of view. **Awd** is just an optimization code, and correct mathematical programming results do not necessarily imply acceptable results. The results will be acceptable only if the problem is correctly formulated. The user should keep in mind that a mathematical program solver like LSSOL will minimize the *given* objective function subject to the *given* constraints. Thus, any feature of concern must be included either as a constraint or as an objective. Otherwise, the code will not "care" about such a feature. Ignoring some features or merely specifying a feasibility problem may very well yield an array space factor that is not "acceptable."

7.5 Awd Error Handling

All syntax errors are caught, but the resulting error messages are sometimes hard to connect to the actual error in the source code. No effort was made to accurately report syntax errors or recover from them. If **awd** produces a long stream of syntax errors, look only at the first one that it prints. Fix this and then re-run **awd**.

8 MATLAB

PRO-MATLAB is not a requirement for using **awd**, but it may simplify specifying the array structure and simulating the produced results. Thus, some supporting C-programs and PRO-MATLAB scripts are available to the user. In the following, we will detail the description of each of these scripts.

8.1 Transforming from PRO-MATLAB to Awd

MATLAB is a convenient tool for setting the array description. The following support utilities are available for converting MATLAB format into ASCII format acceptable to **awd**.

- *m2awd*: is a C-program that takes a character string, say "*filename*," as its argument, searches for variables x, y, z, p, and w in a file called "*filename*.`mat`," and writes whichever of these variables it finds to corresponding ASCII files called "*filename*.`x`," "*filename*.`y`," "*filename*.`z`," "*filename*.`p`," and "*filename*.`w`" respectively. These files then contain the x, y, and z locations of the antenna array elements, the noise power of the elements p, and the weight factors w written in the same format that **awd** accepts in its *array description* section. These ASCII files can be conveniently included in the source code of the AWDSL program using the "`#include`" facility of the C preprocessor.

- *mat_2_awd*: is a PRO-MATLAB script that can be run while using MATLAB and it allows the transformation from MATLAB to **awd** to be handled internally. This script looks for a MATLAB ASCII variable called "`file_name`" whose value is, say, '*filename*'. The script then saves the x, y, z, p, and w existing variables to "*filename*.`mat`," and runs *m2awd* to produce the required ASCII files.

8.2 Transforming from Awd to PRO-MATLAB

The following utilities for converting from **awd** output format to MATLAB format are also available.

- *awd2m*: is a C-program that takes a character string, say "*filename*," as its argument, searches for ASCII files called "*filename*.`x`," "*filename*.`y`," "*filename*.`z`," "*filename*.`p`," and "*filename*.`w`." These files must contain the x, y, and z locations of the antenna array elements, the noise power of the elements p, and the weight factors w written in the same format that **awd** accepts in its *array description* section. It then stores the values corresponding to whichever of these files it finds as variables x, y, z, p, and w, respectively, in a file called "*filename*.`mat`." This file can be then conveniently loaded to MATLAB.

- *awd_2_mat*: is a PRO-MATLAB script that can be run while using MATLAB. This script allows the transformation from MATLAB to **awd** to be handled internally.

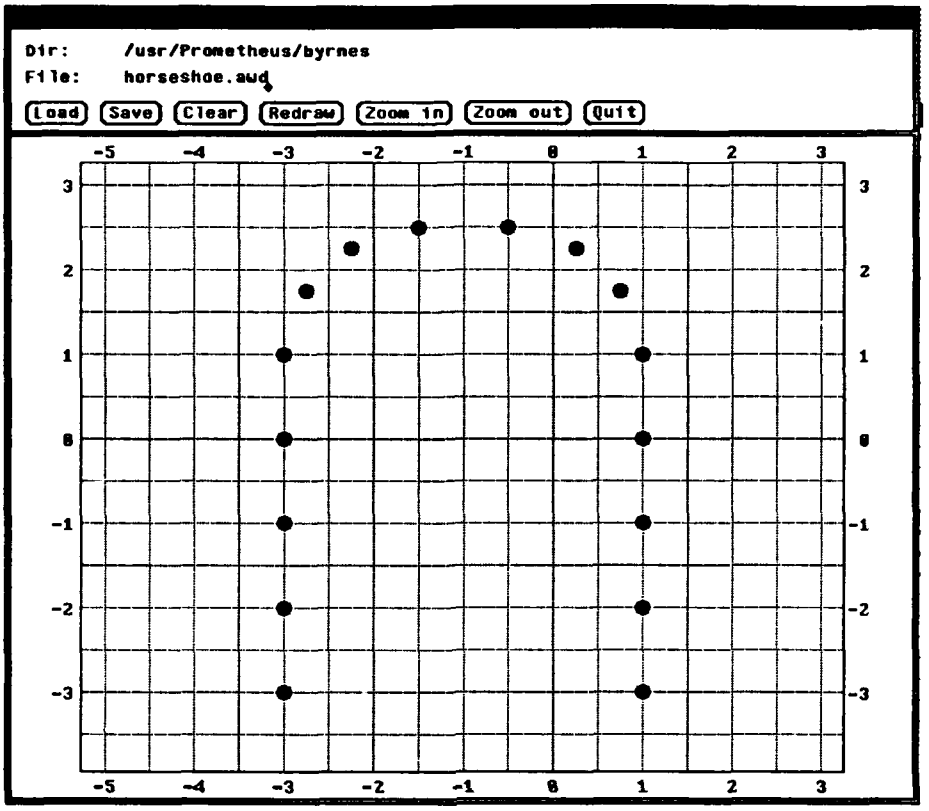


Figure 2: Sample *layouttool* window

This script looks for a MATLAB ASCII variable called "`file_name`" whose value is, say, '*filename*'. The script then runs the program *awd2m* to create a file called "*filename*`.mat`" that contains the x, y, z, p, and w variables. *Awd_2_mat* then loads the above "*filename*`.mat`" file to MATLAB.

8.3 Graphical Layout of Array Elements

The vector handling capabilities built into Matlab provide a convenient means of specifying regularly shaped antenna arrays (e.g., linear arrays, circular arrays, or cylindrical arrays). However, experimentation with more irregular array shapes can be cumbersome. The following support utilities are available to design antenna arrays interactively:

- *layouttool*: is a C-program that takes a character string, say "filename," as its argument, and searches for an ASCII file by that name. This file must contain the number of array elements, the x, y, and z locations, the noise power p, and the weight factors w of the antenna array elements, written in the same format that *awd* accepts in its

array description section. If no input file argument is given, *layouttool* will start up with an empty array. *layouttool* starts up a Suntools window, providing the following functionality:

- *placing new elements*: click the left mouse button on a free space inside the drawing area
- *moving elements*: click at an element with the left mouse button, move the mouse to the desired location, and release the mouse button
- *deleting elements*: click the middle mouse button at the element to be deleted
- *editing element parameters numerically*: click the right mouse button at the element to view a pop-up window containing the element parameters, fill in the appropriate fields, and acknowledge by clicking the *ok* button
- *loading an antenna array specification*: fill in the *dir* and *file* fields and click the *load* button
- *saving an antenna array specification*: fill in the *dir* and *file* fields and click the *save* button
- *clear all elements*: click the *clear* button
- *redraw screen* (in case the display got corrupted): click the *redraw* button
- *increase resolution*: click the *zoom in* button
- *decrease resolution*: click the *zoom out* button
- *quit layouttool*: click the *quit* button (make sure to save the array specification before quitting)
- *scroll to the right*: click at the scale to the left of the drawing area
- *scroll to the left*: click at the scale to the right of the drawing area
- *scroll down*: click at the scale at the top of the drawing area
- *scroll up*: click at the scale at the bottom of the drawing area

- *layout*: is a PRO-MATLAB script that can be run while using MATLAB. This script saves the Matlab variables n, x, y, z, p, and w (number of elements, x, y, z positions, noise power, and element weights, respectively) into the file "array.mat", converts it into an *awd* file "array.awd", and starts up the *layouttool* with that file as an input. Upon quitting the *layouttool*, the *layout* script converts "array.awd" back into "array.mat", and loads that file into MATLAB.

8.4 Simulating Awd Results with PRO-MATLAB

The following functions are available:

- *get_s[w,x,y,z,θ,φ,f,k_{ref}]*: is a PRO-MATLAB *function* that can be run while using MATLAB. This function calculates the antenna array space factor S (as described above) at azimuth angle θ (in degrees), elevation angle ϕ (in degrees) and (normalized) frequency f with weight factors w and geometry described by x, y, and z. Pre-steering information (if any) is given in k_{ref}. If this last input is omitted it will be defaulted to zero. The result will be a complex number.

  Note that θ, ϕ, and f can be vectors. In this case, the output of the function will be a (complex) matrix of `length(`f`)*length(`ϕ`)` rows and `length(`θ`)` columns. The first `length(`f`)` rows will contain function values obtained using the first element of ϕ, and so on.

- *db_plot[what,w,x,y,z,θ,φ,f,k_{ref}]*: is a PRO-MATLAB *function* that can be run while using MATLAB. This function assumes that θ and ϕ are given in degrees. The function first calls *get_s[w,x,y,z,θ,φ,f,k_{ref}]* to calculate the antenna array space factor S.

  If "what" is set to 1, f should be a scalar. If "what" is set to 2, θ should be a scalar. Otherwise, ϕ should be a scalar. The function evaluates the absolute value of the S matrix in decibels and plots it as a three-dimensional object (using PRO-MATLAB's `mesh` function). If "what" is set to 1, such object will be a plot of the magnitude of the array in decibels versus θ and ϕ. If "what" is set to 2, such object will be a plot of the magnitude of the array in decibels versus f and ϕ. Otherwise, the three-dimensional object will be a plot of the magnitude of the array in decibels versus f and θ.

  If one (but not both) of the vectors that should enter the 3-D plot is actually a scalar, a two dimensional plot of the magnitude of S in decibels as a function of the remaining vector entry will result.

  The function will return the output of `get_s`.

- *polar_plot[w,x,y,z,θ,φ,f,k_{ref}]*: is a PRO-MATLAB function that can be run while using MATLAB. This function requires that ϕ and f be scalars. The function calls *get_s* to calculate the value of the space factor S. The function then plots the magnitude of the space factor in polar coordinates.

- *polar_db_plot[w,x,y,z,θ,φ,f,k_{ref}]*: is a PRO-MATLAB function that can be run while using MATLAB. This function is the same as *polar_plot* except that the plots will be in decibels. It should be noted here that the value of the magnitude of S, will be scaled to overcome certain difficulties arising from the use of MATLAB.

- *plot_array(x,y,z, what)*: is a PRO-MATLAB function that can be run while using MATLAB. If "what" is set to 1, the function plots the projection of the array geometry on the x-y plane. If "what" is set to 2, the the projection of the array geometry on the x-z plane will be plotted. Finally, if "what" is set to 3, the function plots the projection of the array geometry on the y-z plane.

9 EXAMPLES

9.1 Linear Array of Narrowband Antenna Elements

In this section, **awd** is used to solve an antenna design example for a linear array of 20 narrowband elements equally spaced along the y-axis between 0 and 9.5λ. The array is assumed to lie in a plane parallel to the $x-y$ plane at $z = 2\lambda$. It is assumed that the element noise power is unity for all elements.

The design objective is to minimize the combined array noise power. The array space factor is to have at least a unit gain at -15 degrees, with no sidelobe gain of more than -30 dB outside a 20 degree sector centered at the target angle. It is also desired to have a -50 dB null in the 40 degree sector from -75 to -35 degrees.

A source file for such a program is as follows. Note that the (normalized) frequency index is omitted since array elements are assumed to be narrowband.

```
/*
Linear array of 20 elements spaced at 0.5 wavelength along the y-axis.
*/

/* basic definitions: */
#define        TARGET          -15
#define        NULL            -55
#define        BEAM_WIDTH       20
#define        NULL_WIDTH       40
#define        NULL_GAIN       -50
#define        SIDE_GAIN       -30

array_description{

        n_elements = 20; /* the number of antenna elements in the array */

        for i = 0 to n_elements-1 do {
                element_loc_x(i) = 0.0;
                element_loc_y(i) = i*0.5;
                element_loc_z(i) = 2;
                element_pwr(i) = 1.0;
        }
}

minimize{
        noise_pwr;
}
```

```
subject_to
{
/* target angle TARGET as defined above */
        re(S(TARGET)) >= 1;

/* a null centered at NULL degrees and of width NULL_WIDTH */
        max_mag_S(NULL-NULL_WIDTH/2,NULL+NULL_WIDTH/2,100) <= inv_dB(NULL_GAIN);

/* no sidelobe has gain hi  r than SIDE_GAIN dB */
        max_mag_S(-90, TARGET-BEAM_WIDTH/2) <= inv_dB(SIDE_GAIN);
        max_mag_S(TARGET+BEAM_WIDTH/2, 90) <= inv_dB(SIDE_GAIN);
}
```

The .out file for such a program looks as follows.

```
                        awd  version 3.0
                  copyright Prometheus Inc.  1988

element    x-loc      y-loc      z-loc      noise_pwr      complex weight
0          0          0          2          1          0.01317    0.008598
1          0          0.5        2          1          0.01626    0.01105
2          0          1          2          1          -0.003668  0.02777
3          0          1.5        2          1          -0.02633   0.02438
4          0          2          2          1          -0.04747   -0.004941
5          0          2.5        2          1          -0.03337   -0.04614
6          0          3          2          1          0.01016    -0.06534
7          0          3.5        2          1          0.06185    -0.04119
8          0          4          2          1          0.07566    0.01723
9          0          4.5        2          1          0.04208    0.06901
10         0          5          2          1          -0.02257   0.07739
11         0          5.5        2          1          -0.06809   0.03611
12         0          6          2          1          -0.06931   -0.02297
13         0          6.5        2          1          -0.02624   -0.06
14         0          7          2          1          0.02151    -0.05211
15         0          7.5        2          1          0.04692    -0.01678
16         0          8          2          1          0.03172    0.01725
17         0          8.5        2          1          0.008442   0.02679
18         0          9          2          1          -0.01457   0.01529
19         0          9.5        2          1          -0.01105   0.01084

constraint                    value   lo_bnd  up_bnd    lambda    status
```

```
re(S(-15))                              1        1        1e+10     0.1296    lb
max_mag_S(-75,-35,100)(90)(1)        0.00329 0        0.003162 -0.5443    ub
                        Above constraint attains bounds at:
                -69.4       90          1           0.003217   -0.02565
                -68.6       90          1           0.003241   -0.009777
                -59.8       90          1           0.003071   -0.03465
                -59.4       90          1           0.003122   -0.02704
                -51.4       90          1           0.003173   -0.07514
                -50.6       90          1           0.003138   -0.01113
                -44.6       90          1           0.003107   -0.09142
                -43.8       90          1           0.003176   -0.0215
                -38.6       90          1           0.003175   -0.1272
                -38.2       90          1           0.003228   -0.01943
                -35         90          1           0.00329    -0..014
max_mag_S(-90,-25,100)(90)(1)        0.03272 0        0.03162  -0.03585  ub
                        Above constraint attains bounds at:
                -27.6       90          1           0.03046    -0.01309
                -26.95      90          1           0.03272    -0.005512
                -26.3       90          1           0.03042    -0.01725
max_mag_S(-5,90,100)(90)(1)          0.0329  0        0.03162  -0.1475   ub
                        Above constraint attains bounds at:
                -5          90          1           0.03054    -0.006633
                -4.05       90          1           0.03224    -0.0285
                -0.25       90          1           0.03176    -0.01261
                0.7         90          1           0.03113    -0.01891
                5.45        90          1           0.0329     -0.02123
                11.15       90          1           0.03171    -0.01002
                12.1        90          1           0.0304     -0.002987
                16.85       90          1           0.03081    -0.00645
                17.8        90          1           0.0325     -0.00292
                23.5        90          1           0.03077    -0.006671
                30.15       90          1           0.03079    -0.004516
                31.1        90          1           0.03134    -0.0007145
                37.75       90          1           0.03052    -0.004529
                46.3        90          1           0.03211    -0.003496
                47.25       90          1           0.03065    -0.001102
                55.8        90          1           0.03109    -0.00217
                57.7        90          1           0.03041    -0.003682
                71.95       90          1           0.03047    -0.005647
                72.9        90          1           0.03049    -0.004698
found strong minimum in 751 iterations
objective function value = 0.061177
```

```
total noise power = 0.0611725
```

The next three graphs contain plots of the array geometry, the magnitude of the array space factor in polar coordinates and the magnitude of the array space factor in dB versus the incident angle The resulting main lobe 3 dB beam width is 7°.

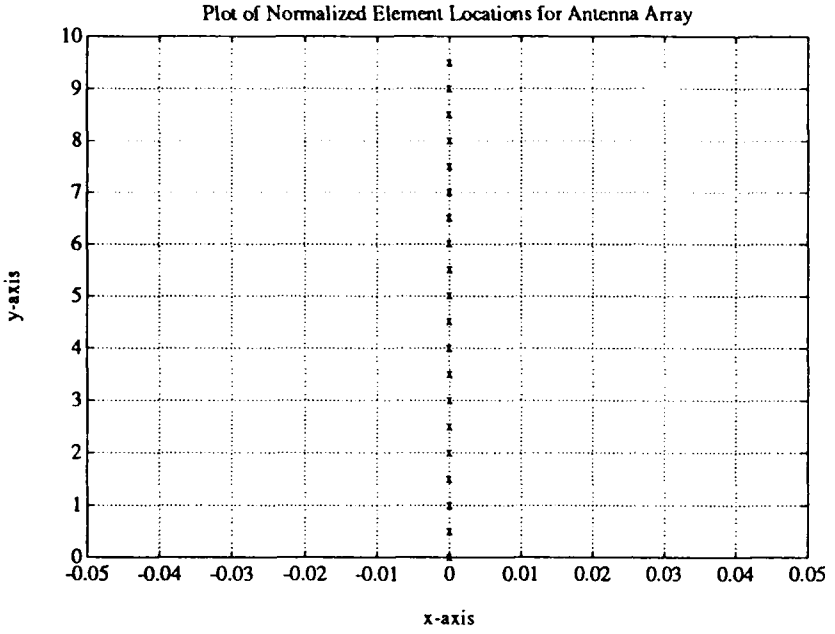


Figure 3: PRO-MATLAB plot of array element locations for example 9.1.

9.2 Linear Array of Wideband Antenna Elements

In this section, **awd** is used to solve an antenna design example for a linear array of 20 wideband elements equally spaced along the y-axis between 0 and 9.5λ. The elements lie in a plane parallel to the $x - y$ plane at $z = 0.5\lambda$. Taking the nominal frequency as the center frequency, the (normalized) frequency band of operation is assumed to be [0.9, 1.1]. It is assumed that the element noise power is unity for all elements.

The design objective is to minimize the combined array noise power. Signals of concern are those that lie in the plane of the array. The array space factor is to have at least unit gain at all band frequencies in the -15 degrees direction, with no sidelobe gain (at any frequency) of more than -20 dB outside a 20 degree sector centered at the target angle. It is also desired to have a -40 dB null in the 40 degree sector from -75 to -35 degrees for all frequencies.

A source file for such a program is as follows.

```
/*
```

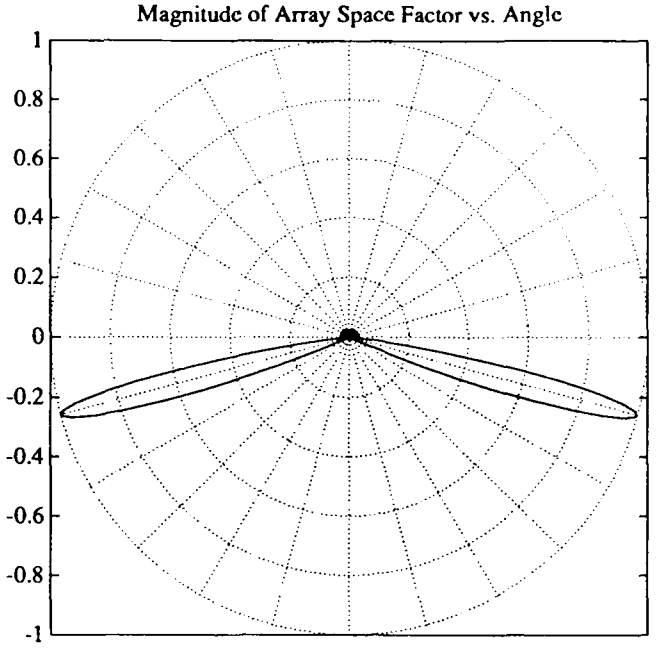


Figure 4: PRO-MATLAB polar plot of magnitude of array space factor for example 9.1.

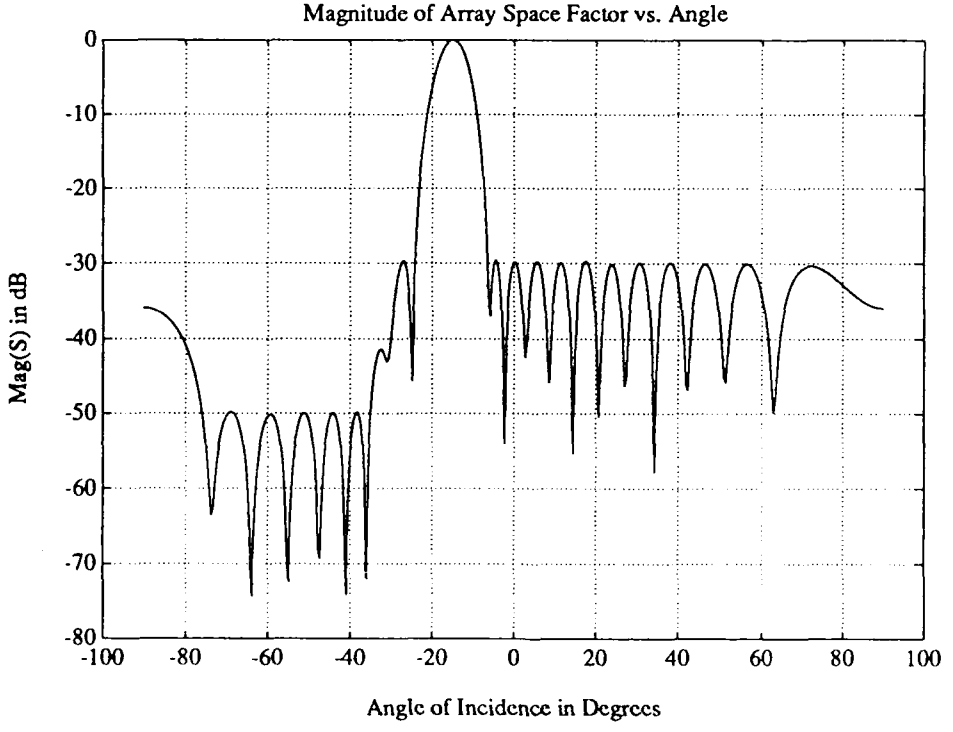


Figure 5: PRO-MATLAB plot of magnitude of array space factor in dB for example 9.1.

```
Linear array of 20 elements spaced at 0.5 wavelength along the y-axis.
*/

/* basic definitions: */
#define        TARGET          -15       /* target direction  */
#define        NULL            -55       /* null direction    */
#define        BEAM_W           20       /* beam width        */
#define        NULL_W           40       /* null sector width */
#define        NULL_G          -40       /* null gain         */
#define        SIDE_G          -20       /* side lobe gain    */
#define        L_FREQ          0.9       /* lower frequency   */
#define        U_FREQ          1.1       /* upper frequency   */

array_description{

        n_elements = 20; /* the number of antenna elements in the array */

        lower_freq = L_FREQ; /* lower bound on operational frequency band */
        upper_freq = U_FREQ; /* upper bound on operational frequency band */

        for i = 0 to n_elements-1 do     {
                element_loc_x(i) = 0.0;
                element_loc_y(i) = i*0.5;
                element_loc_z(i) = 0.5;
                element_pwr(i) = 1.0;
        }
}

minimize{
        noise_pwr;
}

subject_to
{
/* at least unit gain for all frequencies at angle TARGET */
        for f = lower_freq to upper_freq step 0.02 do
                re(S(TARGET,,f)) >= 1;

/* a null centered at NULL degrees and of width NULL_W for all frequencies*/
        max_mag_S(NULL-NULL_W/2,NULL+NULL_W/2,20)()(lower_freq,upper_freq,10)
                                                        <= inv_dB(NULL_G);
```

```
/* no sidelobe has gain higher than SIDE_G dB for any frequency */
        max_mag_S(-90, TARGET-BEAM_W/2, 40)()(lower_freq,upper_freq,10)
                                                    <= inv_dB(SIDE_G);
        max_mag_S(TARGET+BEAM_W/2, 90, 40)()(lower_freq,upper_freq,10)
                                                    <= inv_dB(SIDE_G);
}
```

The resulting .out file for the above source program looks as follows.

```
                        awd  version 3.0
                    copyright Prometheus Inc.  1988
```

element	x-loc	y-loc	z-loc	noise_pwr	complex weight	
0	0	0	0.5	1	0.02534	0.04824
1	0	0.5	0.5	1	0.04649	0.0596
2	0	1	0.5	1	-0.01651	0.05755
3	0	1.5	0.5	1	-0.06919	0.04503
4	0	2	0.5	1	-0.08411	-0.02889
5	0	2.5	0.5	1	-0.04948	-0.08397
6	0	3	0.5	1	0.03232	-0.09897
7	0	3.5	0.5	1	0.08869	-0.05132
8	0	4	0.5	1	0.1017	0.02769
9	0	4.5	0.5	1	0.05035	0.08539
10	0	5	0.5	1	-0.0223	0.09507
11	0	5.5	0.5	1	-0.0767	0.04564
12	0	6	0.5	1	-0.07906	-0.01906
13	0	6.5	0.5	1	-0.03457	-0.06201
14	0	7	0.5	1	0.01689	-0.05426
15	0	7.5	0.5	1	0.04123	-0.01935
16	0	8	0.5	1	0.03069	0.01361
17	0	8.5	0.5	1	0.008726	0.0176
18	0	9	0.5	1	-0.01201	0.01062
19	0	9.5	0.5	1	-0.007842	0.00767

constraint	value	lo_bnd	up_bnd	lambda	status
re(S(-15,90,0.9))	1	1	1e+10	0.1168	lb
re(S(-15,90,0.92))	1.121	1	1e+10	0	
re(S(-15,90,0.94))	1.22	1	1e+10	0	
re(S(-15,90,0.96))	1.293	1	1e+10	0	
re(S(-15,90,0.98))	1.338	1	1e+10	0	
re(S(-15))	1.352	1	1e+10	0	

```
re(S(-15,90,1.02))                          1.336   1       1e+10   0
re(S(-15,90,1.04))                          1.289   1       1e+10   0
re(S(-15,90,1.06))                          1.216   1       1e+10   0
re(S(-15,90,1.08))                          1.118   1       1e+10   0
re(S(-15,90,1.1))                           1       1       1e+10   0.1533    lb
max_mag_S(-75,-35,20)(90)(0.9,1.1,10) 0.0104  0       0.01    -0.7577   ub
                        Above constraint attains bounds at:
                -73         90          0.9         0.01037   -0.01158
                -37         90          0.9         0.01034   -0.07331
                -35         90          0.9         0.01006   -0.06621
                -35         90          0.92        0.009767  -0.03517
                -67         90          0.94        0.01038   -0.06528
                -53         90          0.96        0.009714  -0.05736
                -39         90          0.96        0.0104    -0.07326
                -45         90          0.98        0.009965  -0.04545
                -49         90          1.02        0.009718  -0.03688
                -65         90          1.04        0.009613  -0.07113
                -73         90          1.06        0.009742  -0.04784
                -35         90          1.06        0.0104    -0.06974
                -71         90          1.08        0.009749  -0.008774
                -75         90          1.1         0.0104    -0.02528
                -39         90          1.1         0.009966  -0.07045
max_mag_S(-90,-25,40)(90)(0.9,1.1,10) 0.104   0       0.1     -0.267    ub
                        Above constraint attains bounds at:
                -25         90          0.9         0.104     -0.1223
                -28.25      90          0.92        0.09642   -0.1448
max_mag_S(-5,90,40)(90)(0.9,1.1,10)   0.103   0       0.1     -0.07177  ub
                        Above constraint attains bounds at:
                -5          90          0.9         0.103     -0.01779
                6.875       90          0.9         0.09613   -0.002469
                18.75       90          0.94        0.09612   -0.001434
                -0.25       90          0.96        0.09653   -0.007831
                11.63       90          0.98        0.09614   -0.00393
                -5          90          1.1         0.09781   -0.02107
                -0.25       90          1.1         0.09653   -0.01062
                4.5         90          1.1         0.09654   -0.006615
found strong minimum in 1319 iterations
objective function value = 0.115121
total noise power = 0.115121
```

Figure 5 shows the magnitude of the array space factor (in dB) as a function of the (normalized) frequency and the incident angle (in degrees).

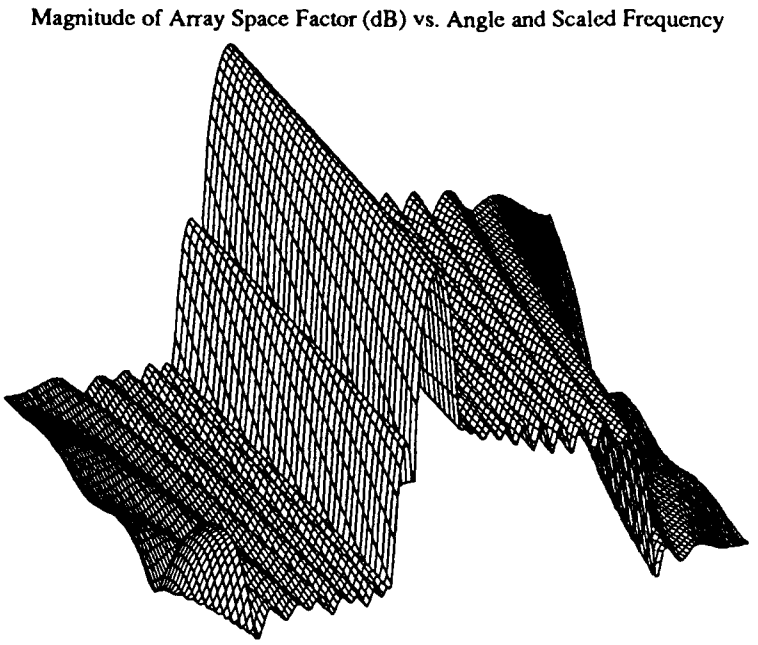


Figure 6: 3-dimensional PRO-MATLAB plot of magnitude of array space factor in dB for example 9.2.

9.3 Cylindrical Array of Narrowband Elements

In this example we show how **awd** can be used to design weights for a 3–D array with narrowband elements. We assume that we have a cylindrical array of 80 elements. Cylinder height is 4λ and its principal axis is the x-axis. The radius of the cylinder is λ. The sensors are grouped in circles of 10 elements each. Circles lie in the y-z plane and are spaced 0.5λ apart along the x-axis. Elements are spaced uniformly on each circle. The following two figures (Fig. 6 and Fig. 7) illustrate the geometry of this array.

Target signals arrive with an azimuthal AOA of 15° and an elevation of 100° (*i.e.* 10° below the x-y plane). It is desired to constrain all sidelobes outside an azimuthal band of width 50° and an elevation band of width 15° centered around the target directions. Sidelobes gains are to be below -12dB.

A possible **awd** source file for such a problem is as follows.

```
/*
A cylindrical array of 80 elements.
*/

#define      N_X_ELT        8
#define      N_RADIAL_ELT  10
#define      HEIGHT         4
```

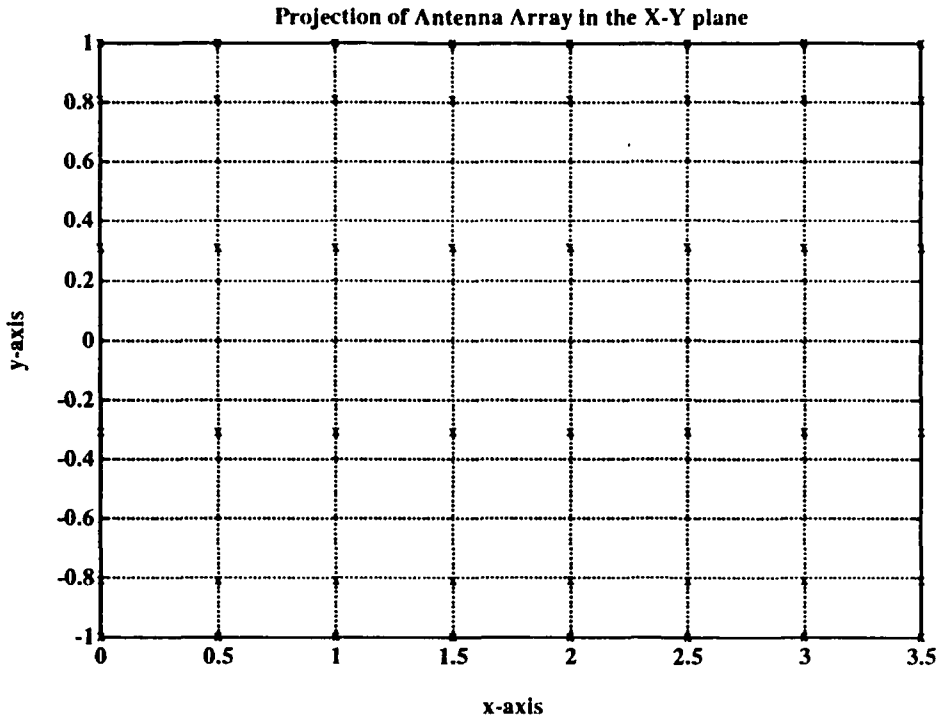


Figure 7: Output of *plot_array*: PRO-MATLAB plot of array element location projected on the x-y plane for example 9.3.

```
#define       SIDE_GAIN    -12
#define       B_WIDTH_A     50
#define       B_WIDTH_E     15

#define       TARGET_A      15
#define       TARGET_E     100

#define       N_AZIM       140
#define       N_ELEV        25

array_description{

      n_elements = 80; /* the number of antenna elements in the array */
      mag_2_lin=2;

      for i=0 to N_X_ELT - 1 do      {
            x = i*HEIGHT/N_X_ELT;
            k = i*N_RADIAL_ELT;
            for j= 0 to N_RADIAL_ELT-1 do      {
```

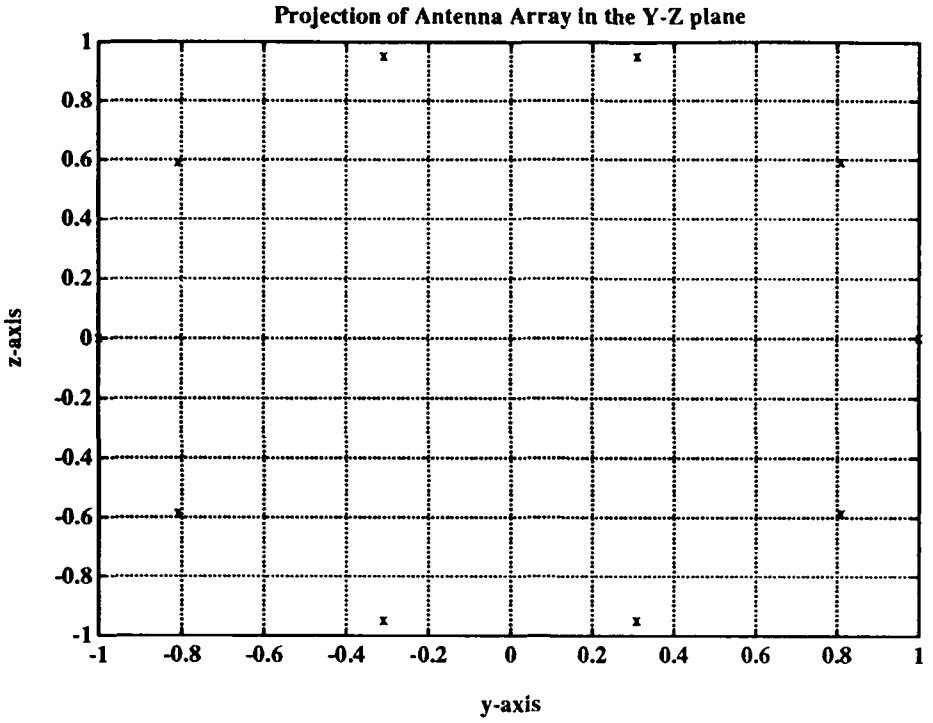


Figure 8: Output of *plot_array*: PRO-MATLAB plot of array element location projected on the y-z plane for example 9.3.

```
                t = j*360/N_RADIAL_ELT;
                element_pwr(k) = 1.0;
                element_loc_x(k) = x;
                element_loc_y(k) = cos(t);
                element_loc_z(k) = sin(t);
                k = k+1;
            }
        }
#include      "ell.w"
}

minimize{
      noise_pwr;
}

subject_to
{
/*  at least unity gain for target direction */
      re(S(TARGET_A, TARGET_E)) == 1;
```

```
/*  constrain every sidelobe outside a sector of band widths
              B_WIDTH_A for azimuth and B_WIDTH_E for elevation */
      max_mag_S(TARGET_A+B_WIDTH_A/2,TARGET_A+360-B_WIDTH_A/2,N_AZIM)
            (0,TARGET_E-B_WIDTH_E/2,N_ELEV)   <= inv_dB(SIDE_GAIN);
      max_mag_S(TARGET_A+B_WIDTH_A/2,TARGET_A+360-B_WIDTH_A/2,N_AZIM)
            (TARGET_E+B_WIDTH_E/2,180,N_ELEV) <= inv_dB(SIDE_GAIN);

/*  help the optimization code by setting some (reasonable) nulls */
      for a=105 to 285 step 20 do      {
            for t = 0 to  60 step 30 do
                  null(a,t);
            for t = 140 to  180 step 20 do
                  null(a,t);
      }
}
```

Plots of the **awd** designed array space factor follows in Fig. 8 and Fig. 9.

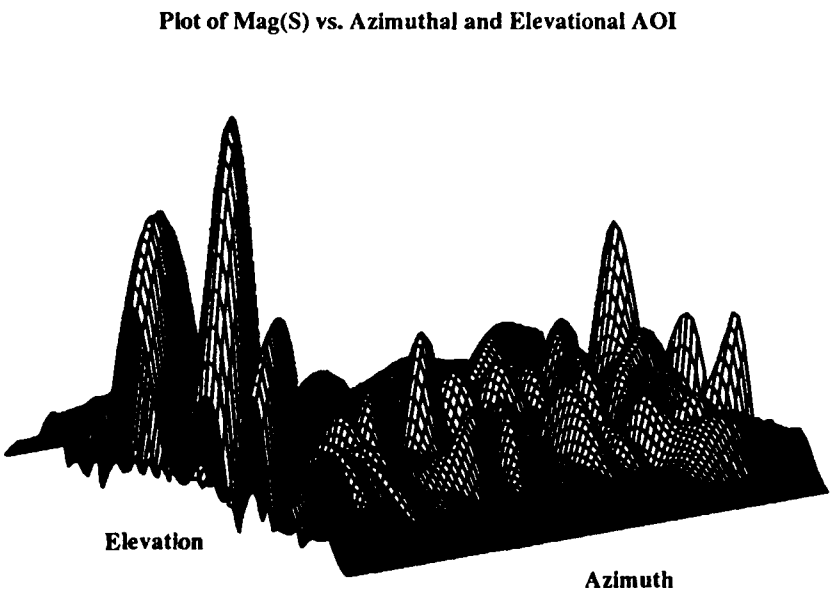


Figure 9: 3-dimensional PRO-MATLAB plot of magnitude of array space factor for example 9.3

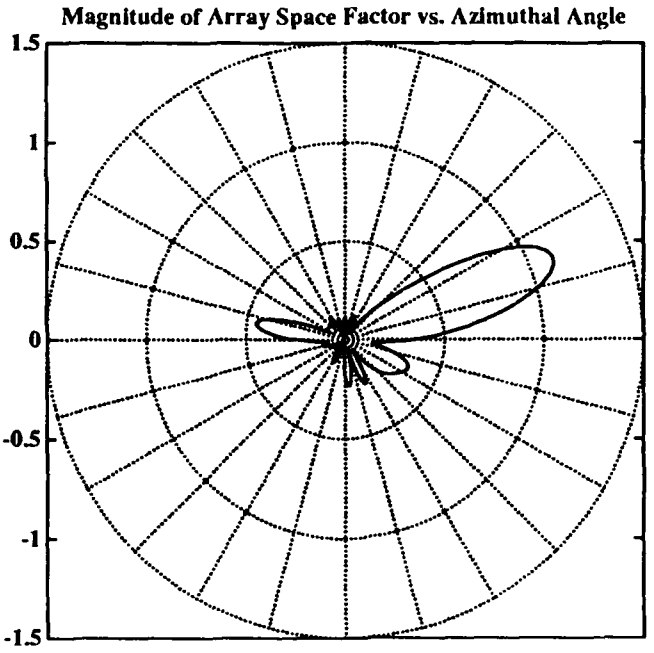


Figure 10: PRO-MATLAB polar plot of magnitude of array space factor vs. azimuth for target elevation angle for example 9.3

10 ACKNOWLEDGMENT

The author is greatly indebted to Stephen Boyd and Richard Roy for their intensive efforts in making this document readable. This document is loosely based on the *Qdes User's Manual*, written by Craig Barratt and Nasser Khraishi.

References

[BBB*88] S. Boyd, V. Balakrishnan, C. Barratt, N. Khraishi, X. Li, D. Meyer, and S. Norman. A new CAD method and associated architectures for linear controllers. *IEEE Trans. Aut. Control*, AC-33(3):268–283, March 1988.

[GHM*86] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for LSSOL (Version 1.0): A FORTRAN Package for Constrained Least-Squares and Convex Quadratic Programming.* Technical Report SOL 86-1, Operations Research Dept., Stanford University, Stanford, California 94305, January 1986.

[GMW81] P. E. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press, London, 1981.

[Joh84] S. C. Johnson. Yacc: Yet another compiler compiler. In *UNIX Programmer's Manual, Supplementary Documents*, University of California, Berkeley, 1984. 4.2 Berkeley Software Distribution.

[KR78] B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice-Hall, 1978.

[LS84] M. E. Lesk and E. Shmidt. Lex - A lexical analyzer generator. In *UNIX Programmer's Manual, Supplementary Documents*, University of California, Berkeley, 1984. 4.2 Berkeley Software Distribution.

[MLBK87] C. Moler, J. Little, S. Bangert, and S. Kleiman. *PRO-MATLAB for Sun Workstations: User's Guide.* The MathWorks, Inc., 1987.

[RWD84] S. Ramo, J. R. Whinnery, and T. Van Duzer. *Fields and Waves in Communication Electronics.* John Wiley and Sons, 2nd edition, 1984.

[Sch43] S. A. Schelkunoff. A mathematical theory of linear arrays. *Bell Syst. Tech. J.*, 22(1):80–107, January 1943.

Prometheus Inc.
Final Report
11 July 1990

Section X
NUSC Array Simulation

PROMETHEUS INC.

Professional Consultants To The Scientific Community

February 1, 1989

Mr. Benjamin Cray
Code 2133
NUSC
New London, CT 06320

Dear Ben,

Enclosed are the results we discussed over the phone yesterday. Figure 1 indicates that I was able to reproduce your results. Based on the *back-lobe* presence, some of the **AWD** results shown minimized max $|S(\theta, f)|$ over the region $[-167.5°, -12.5°] \cup [12.5°, 167.5°]$. It should be obvious from the **AWD** plots when this was done. The mesh plot in Figure 2 was a complex weight design with $f_0 = 170$Hz. Figures 3a and 3b show the results of optimizing over 4 frequency intervals as indicated. Interestingly, the design at the highest frequency interval was significantly poorer than the others. I don't have a real good intuitive explanation for this other than to note that the pre-steering was done at 240Hz for all 4 designs. I am currently working on designs where the steering is done at the mid-point of each interval. The results will follow as they become available.

Note well that the plots over frequency ranges are plots of

$$S(\theta, f) = \sum_{i=1}^{m} w_i e^{j(k(\theta,f)-k(\theta_0,f_0))^T x_i},$$

where $m = 34$ and $\theta_0 = 0°$ for the cases shown, and f_0 is indicated in the figures. This implicitly assumes the ***analytic*** or ***complex*** signal is available (e.g. via Hilbert transform). If, as I am guessing, the current processor configuration simply delays the *real* received signal by $T_0/4$ where $T_0 = 1/f_0$ to generate the ***quadrature*** component, the plots contained herein will bear little (or no) resemblance to those you may have generated with your simulation program that correctly models the processor structure. A simple delay generates the quadrature component for only a single frequency component of the received signal. The overall consequence of such a processing strategy is to create a frequency dependent array manifold from a frequency independent one. Since **AWD** is not currently capable of handling frequency dependent response functions yet, ...

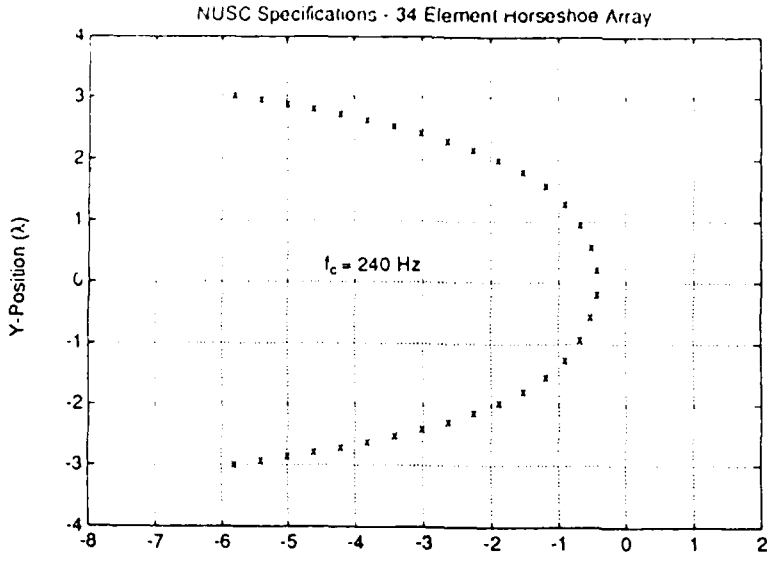
Hope this helps. Please let me know how it is received at NUSC. Also, I'd appreciate finding out who to contact about the upcoming experiment at Seneca Lake.

Sincerly,

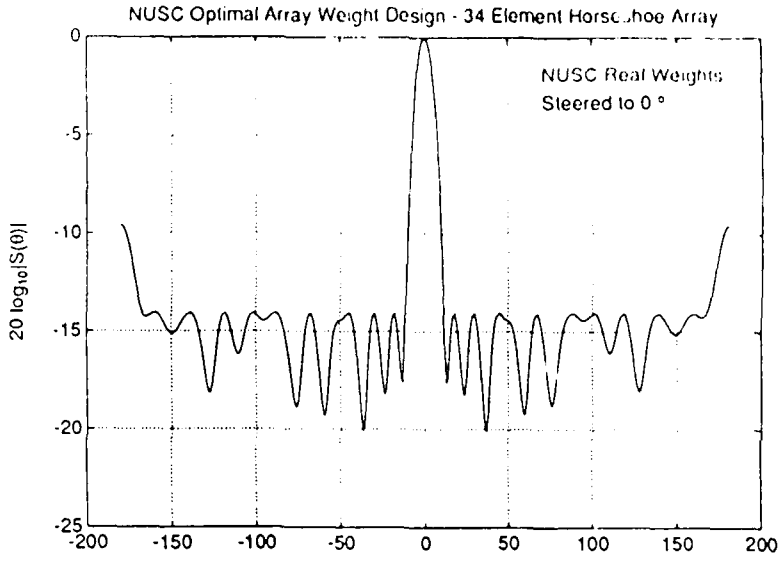
Richard Roy,
Senior Scientist

encl: **AWD** results for 34 element horseshoe array weight design

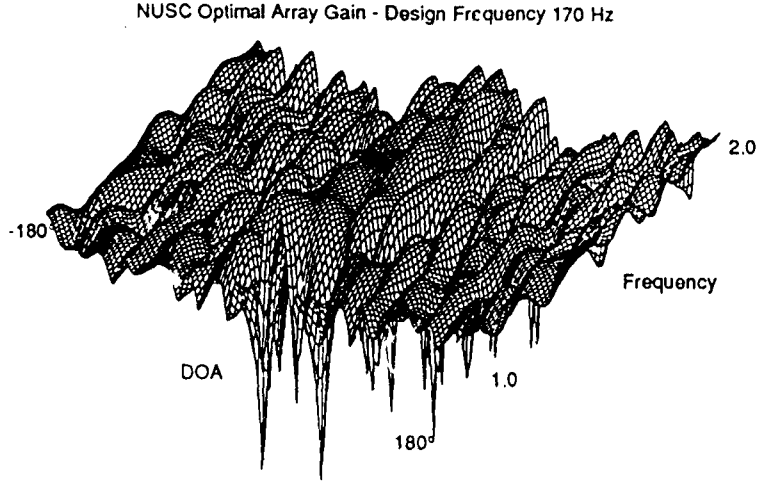
21 Arnold Avenue • Newport, RI. 02840 • 401-849-5369/617-784-2355



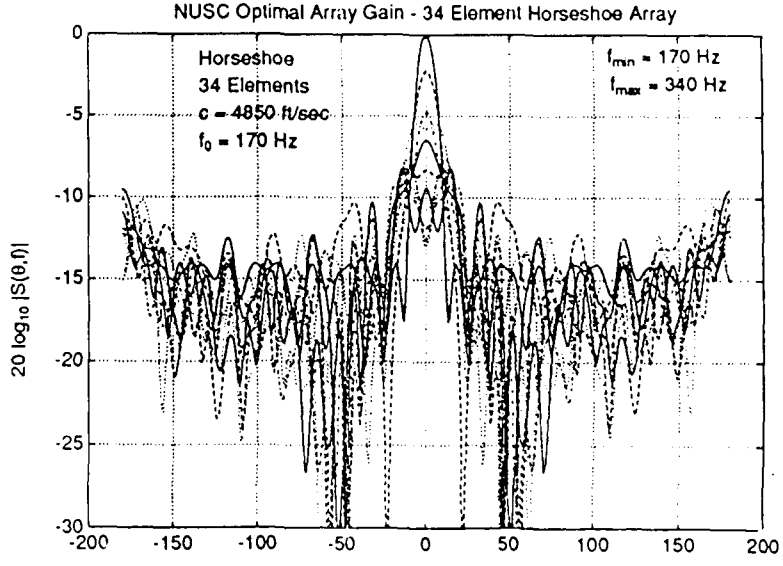
horseshoe32array.ps



NUSCw32steered0.ps

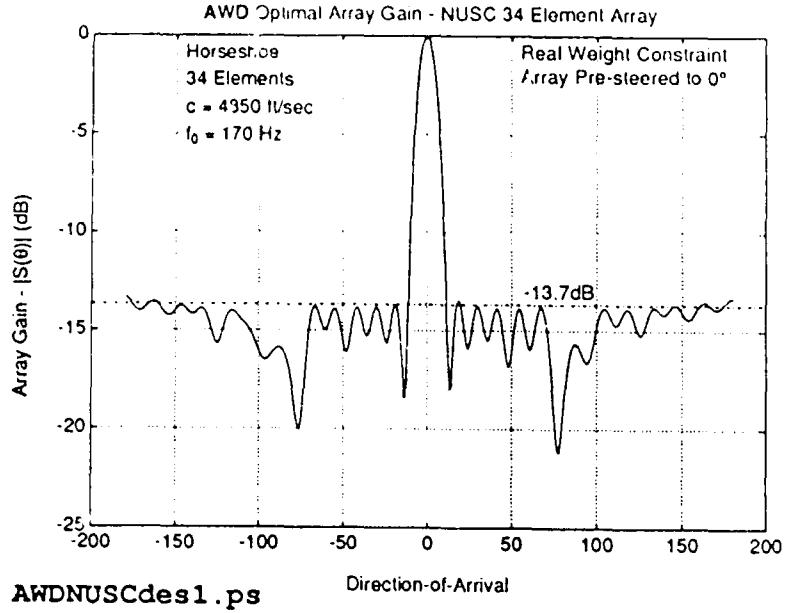


NUSCDOA0mesh1.ps

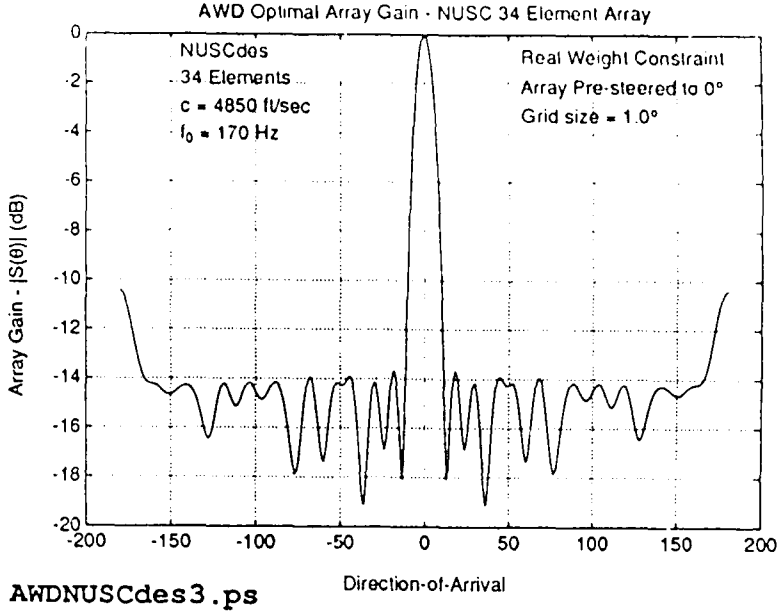


NUSCDOA0allfs.ps

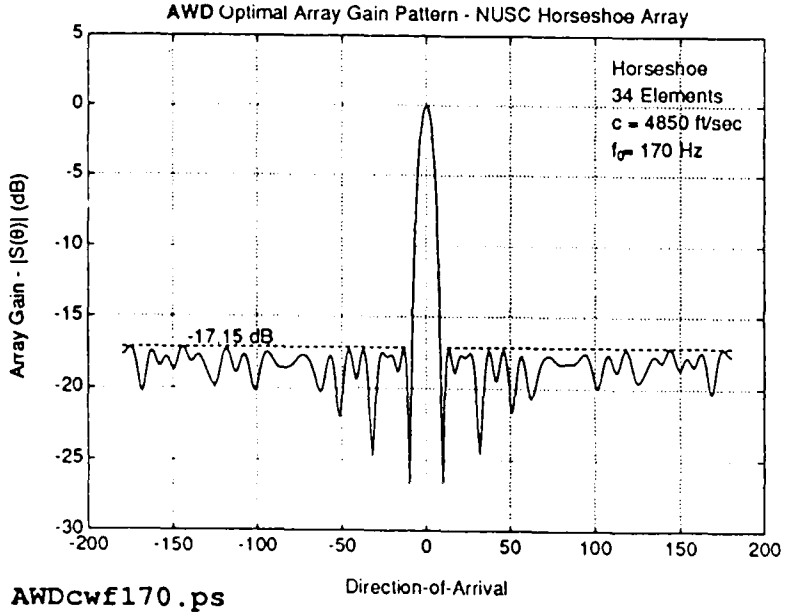
Figure 1



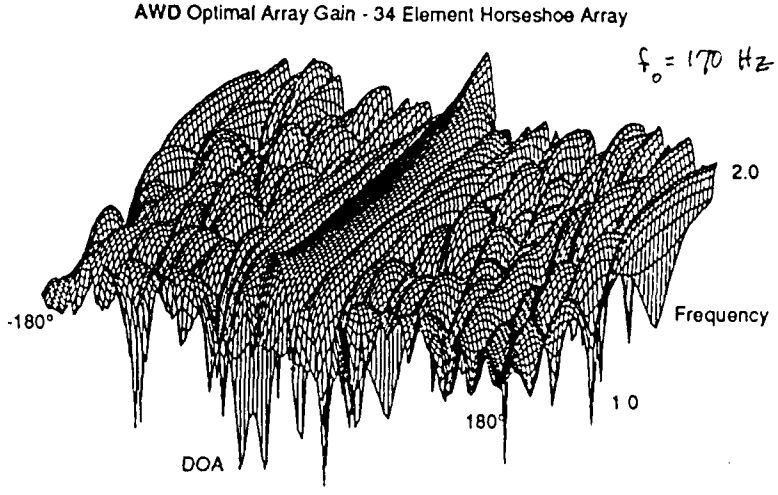
AWDNUSCdes1.ps



AWDNUSCdes3.ps



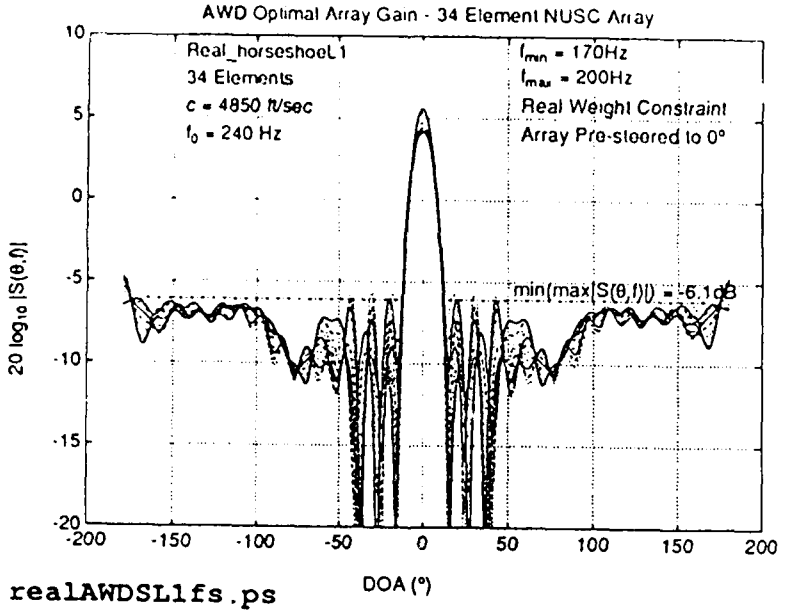
AWDcwf170.ps



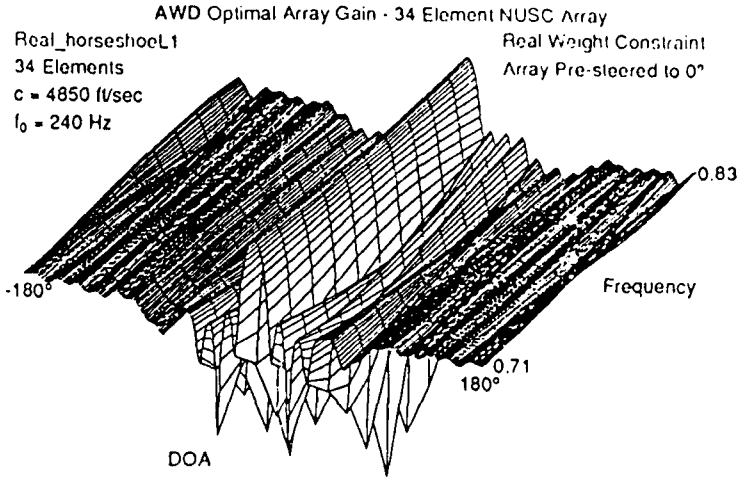
AWDDOA0mesh1.ps

X-3

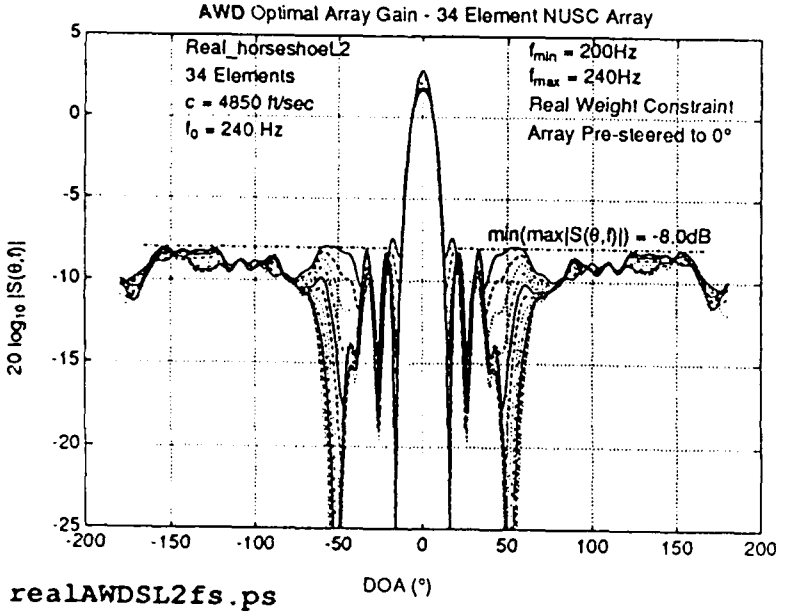
Figure 2



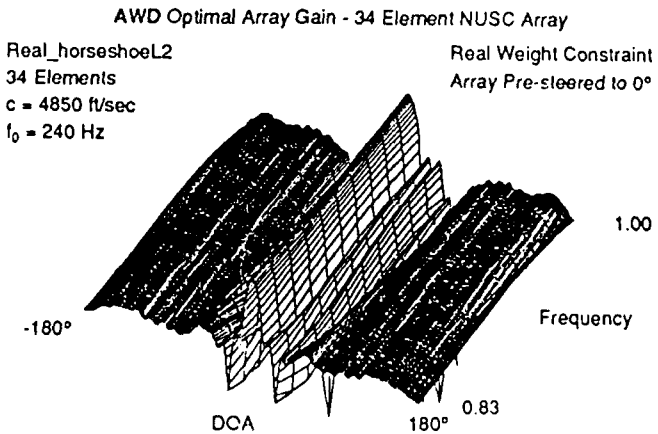
realAWDSL1fs.ps



realAWDSL1mesh1.ps

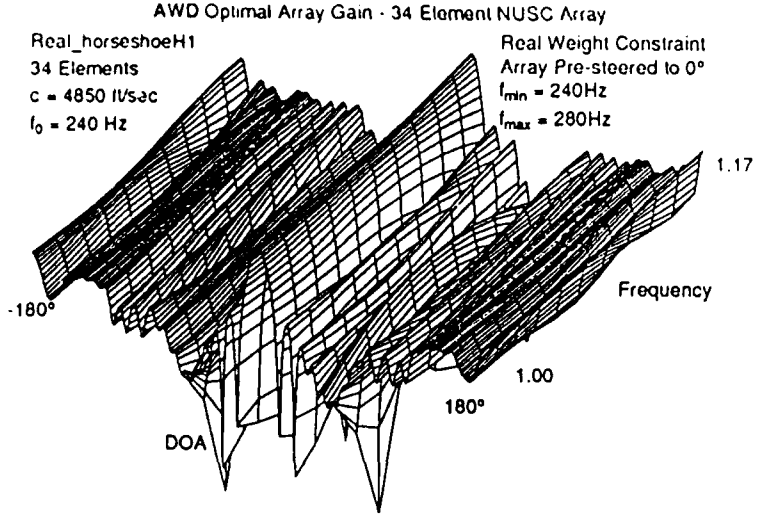


realAWDSL2fs.ps

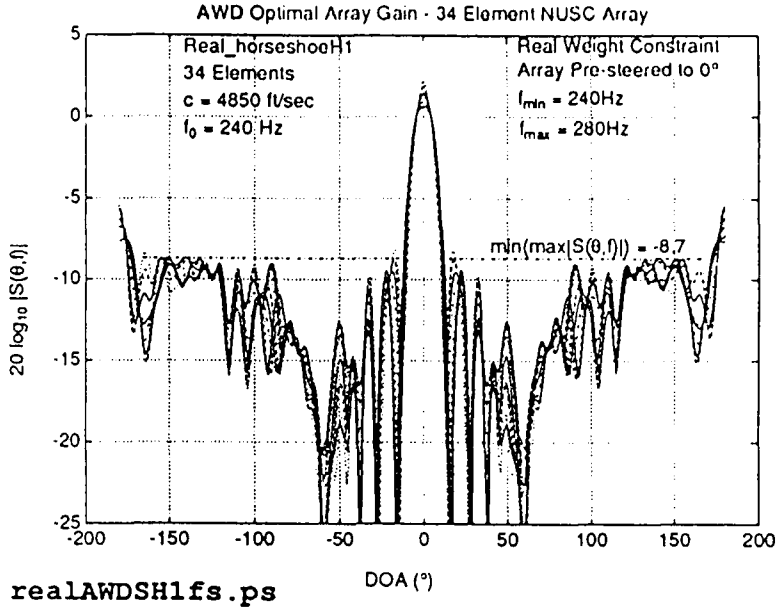


realAWDSL2mesh1.ps

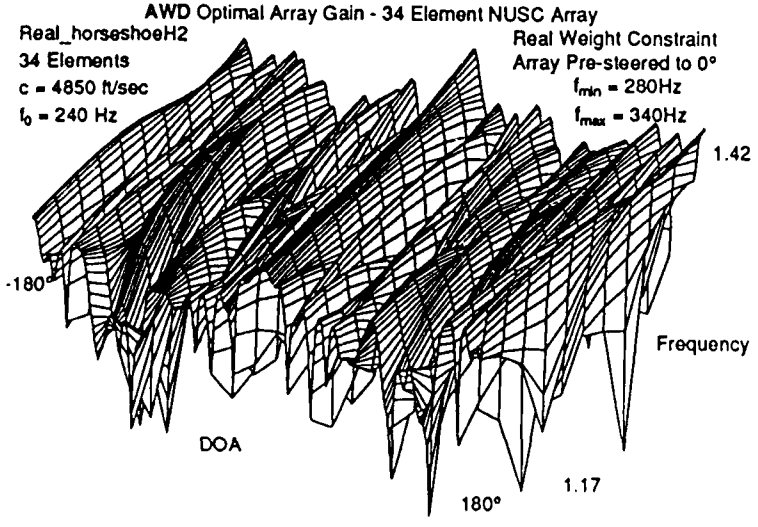
Figure 3a



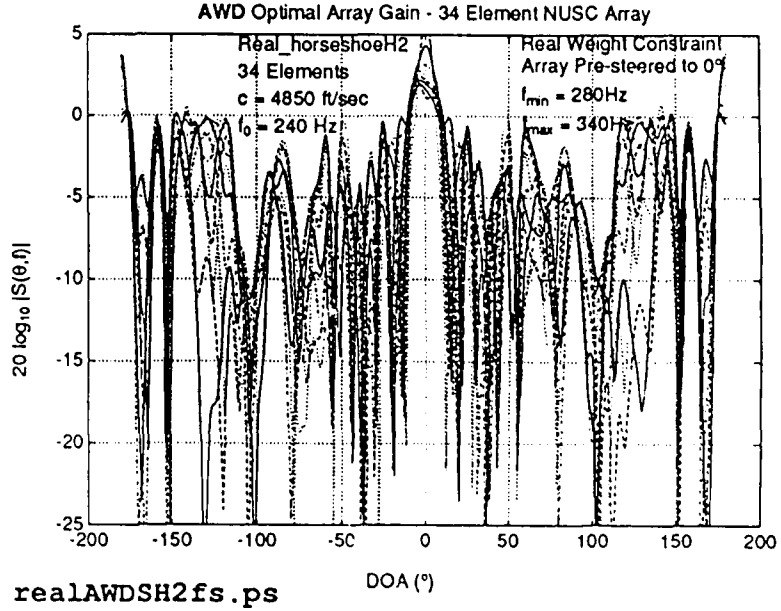
realAWDSH1mesh1.ps



realAWDSH1fs.ps



realAWDSH2mesh1.ps



realAWDSH2fs.ps

Figure 3b

PROMETHEUS INC.

Professional Consultants To The Scientific Community

February 2, 1989

Mr. Benjamin Cray
Code 2133
NUSC
New London, CT 06320

re: **AWD** and the horseshoe array

Dear Ben,

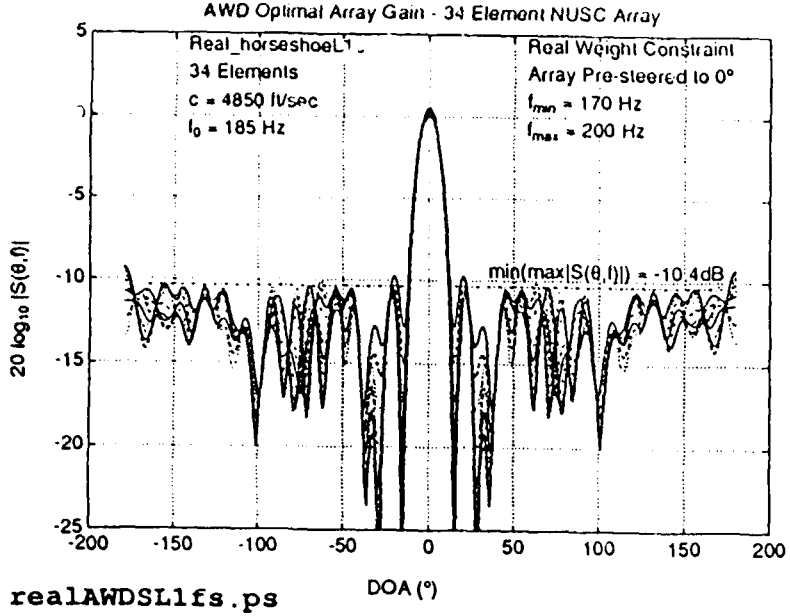
... and here are the results for *centered* design over the four bands. Note the improvement! Again the results assume the *analytic* signal is available over the band.

More as it becomes available,

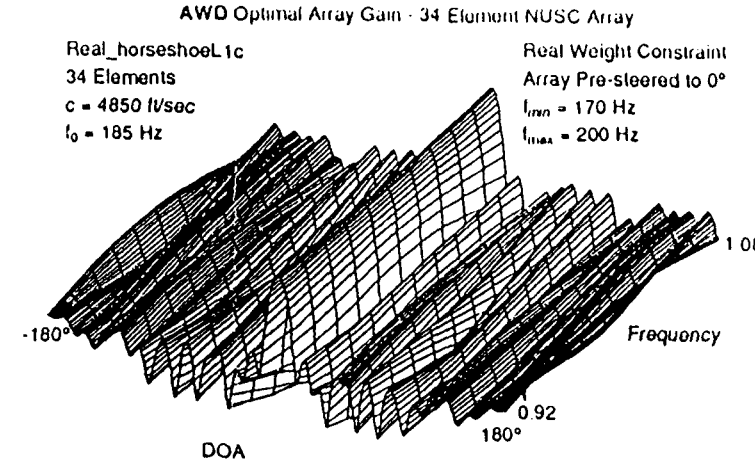
Richard Roy,
Senior Scientist

encl: more **AWD** results for 34 element horseshoe array

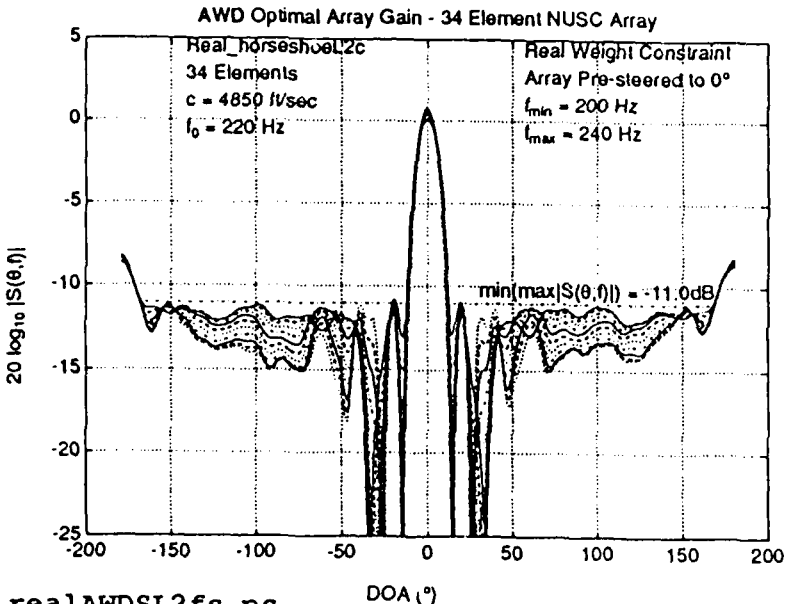
21 Arnold Avenue • Newport, RI. 02840 • 401-849-5389/617-784-2355



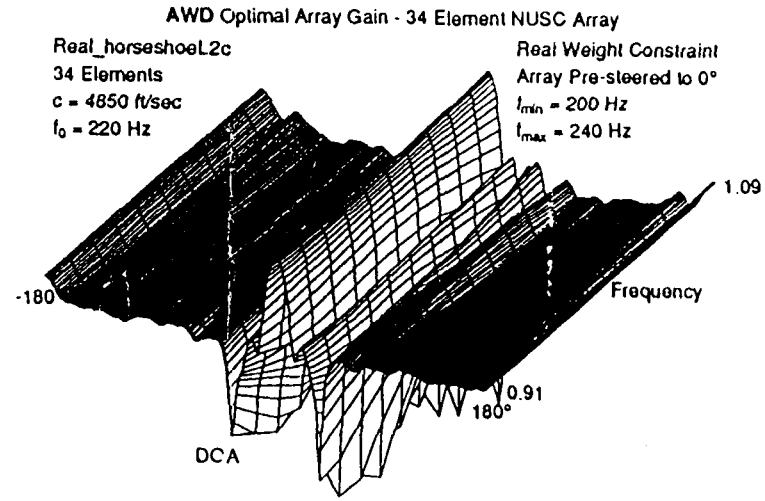
realAWDSL1fs.ps



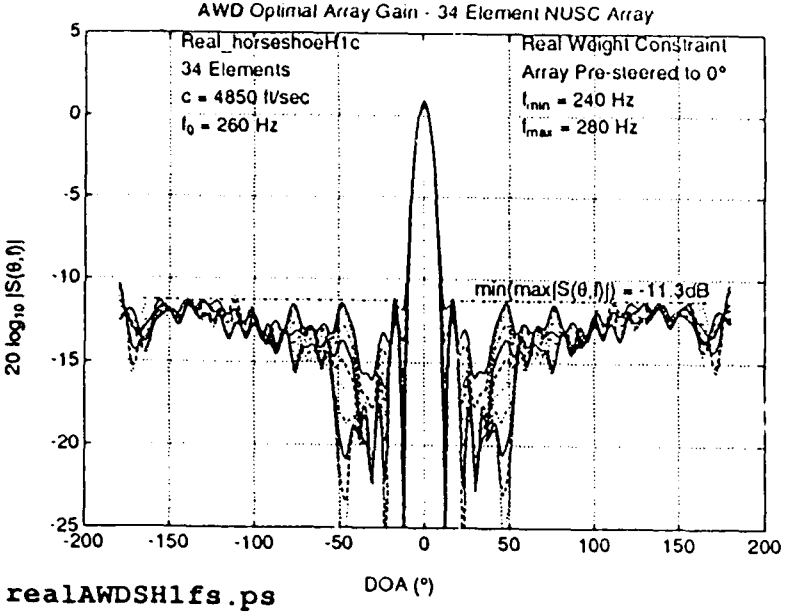
realAWDSL1cmesh1.ps



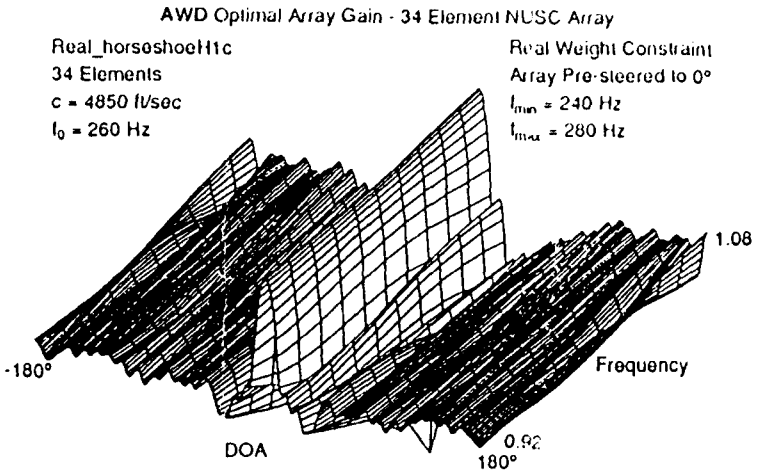
realAWDSL2fs.ps



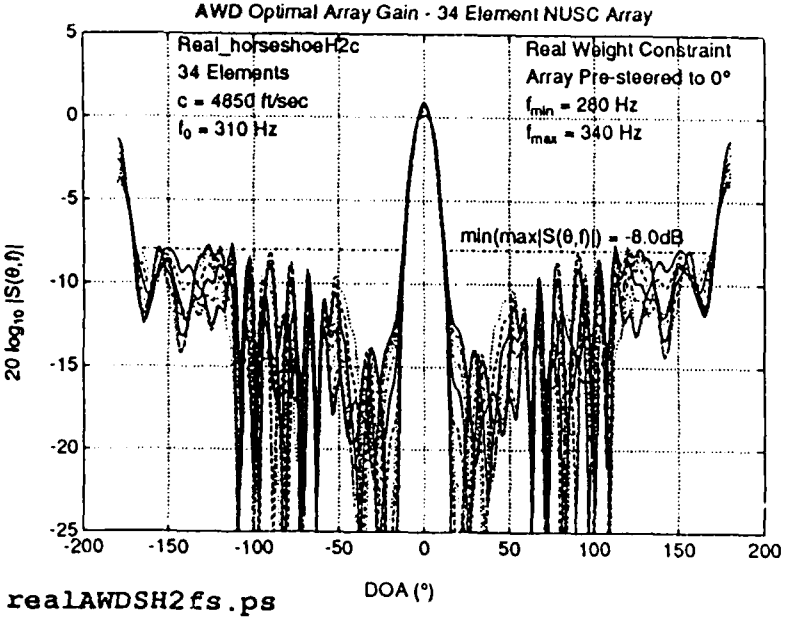
realAWDSL2cmesh1.ps



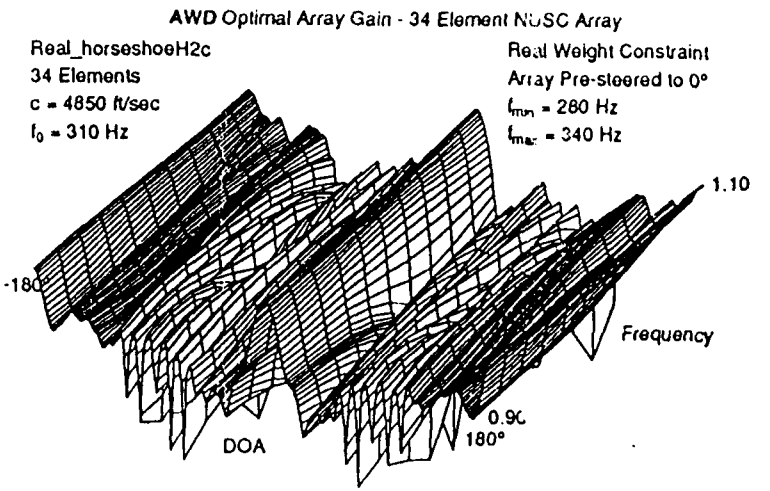
realAWDSH1fs.ps



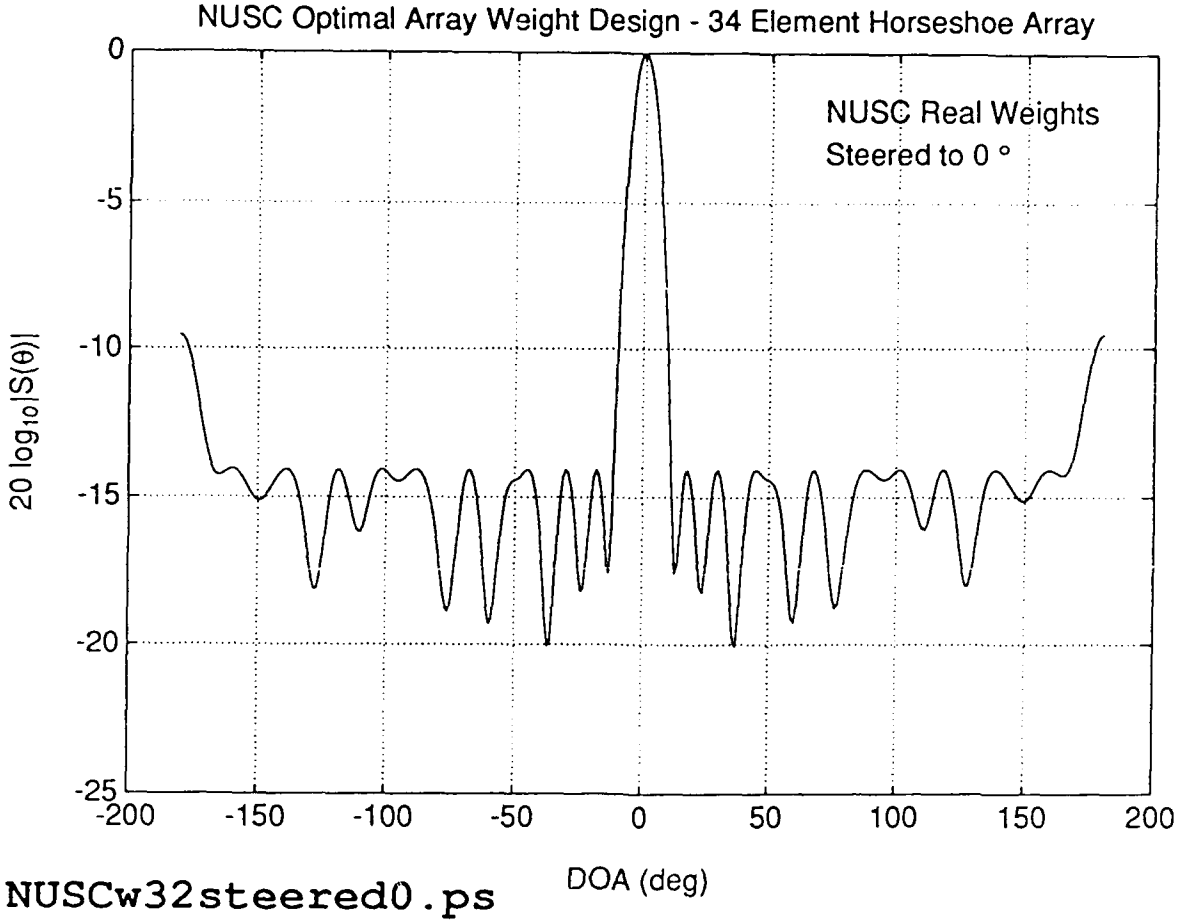
realAWDSH1cmesh1.ps



realAWDSH2fs.ps

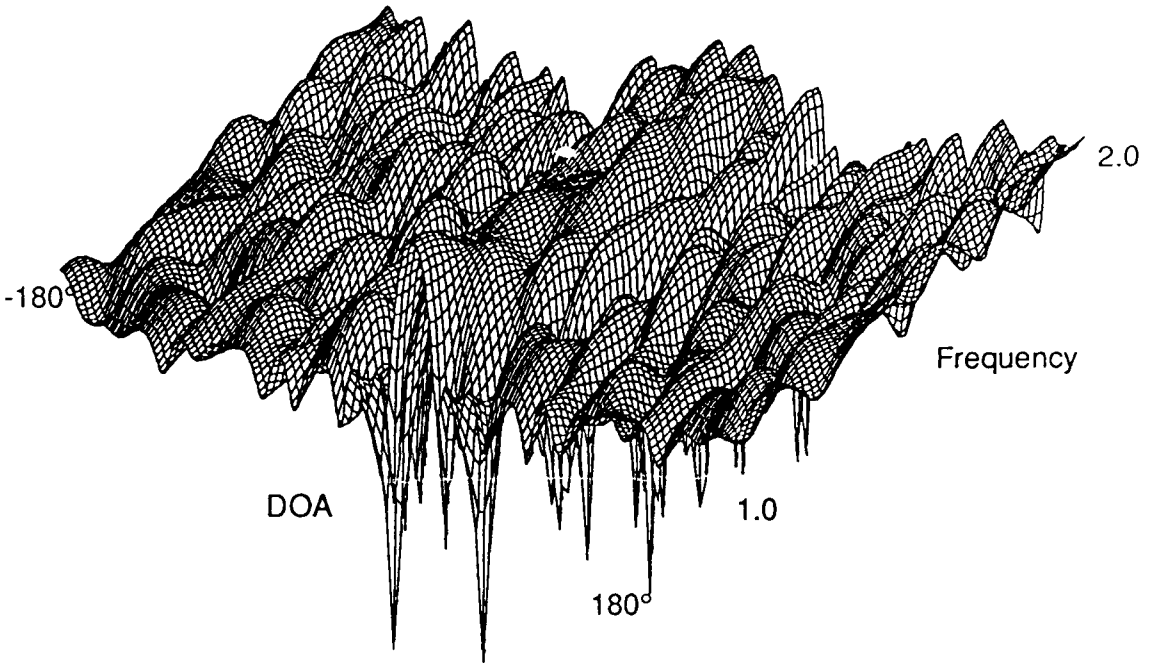


realAWDSH2cmesh1.ps

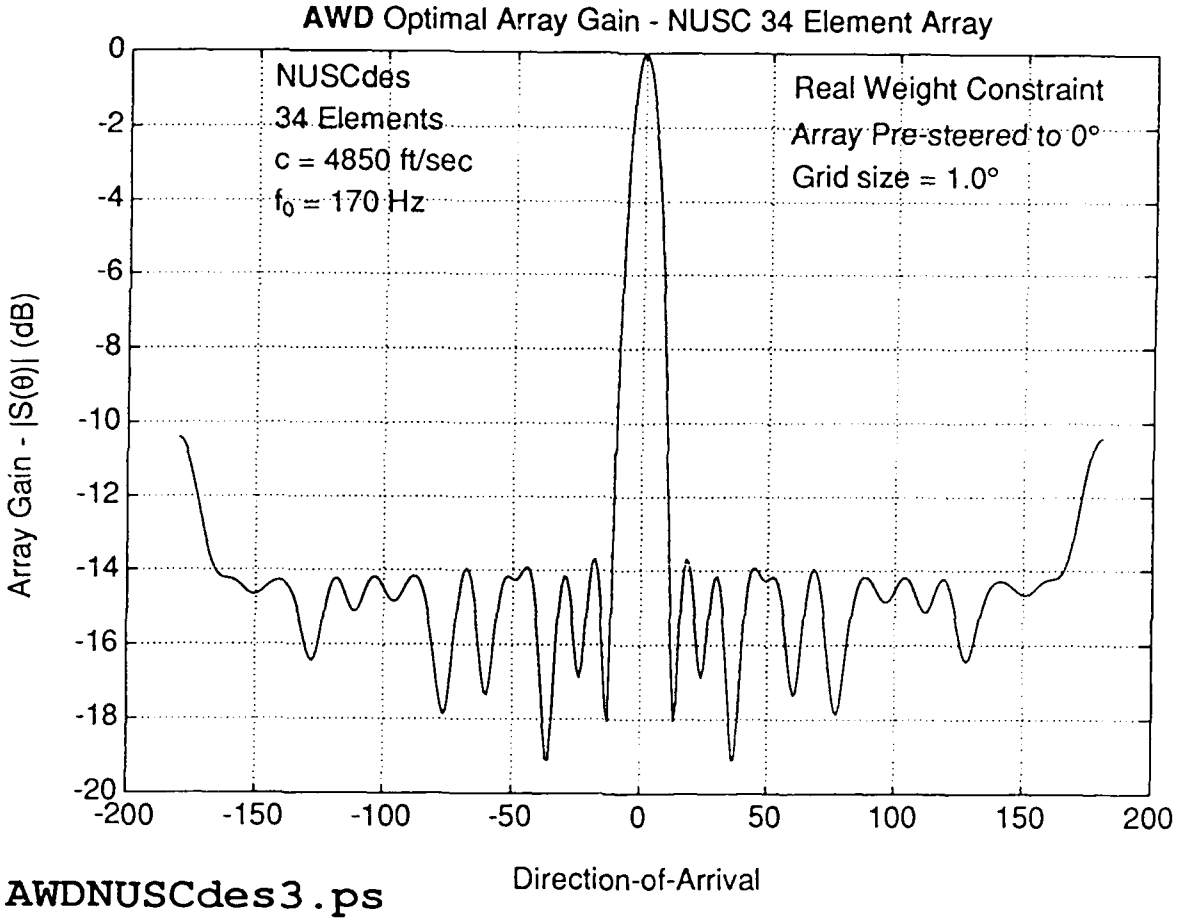


NUSCw32steered0.ps

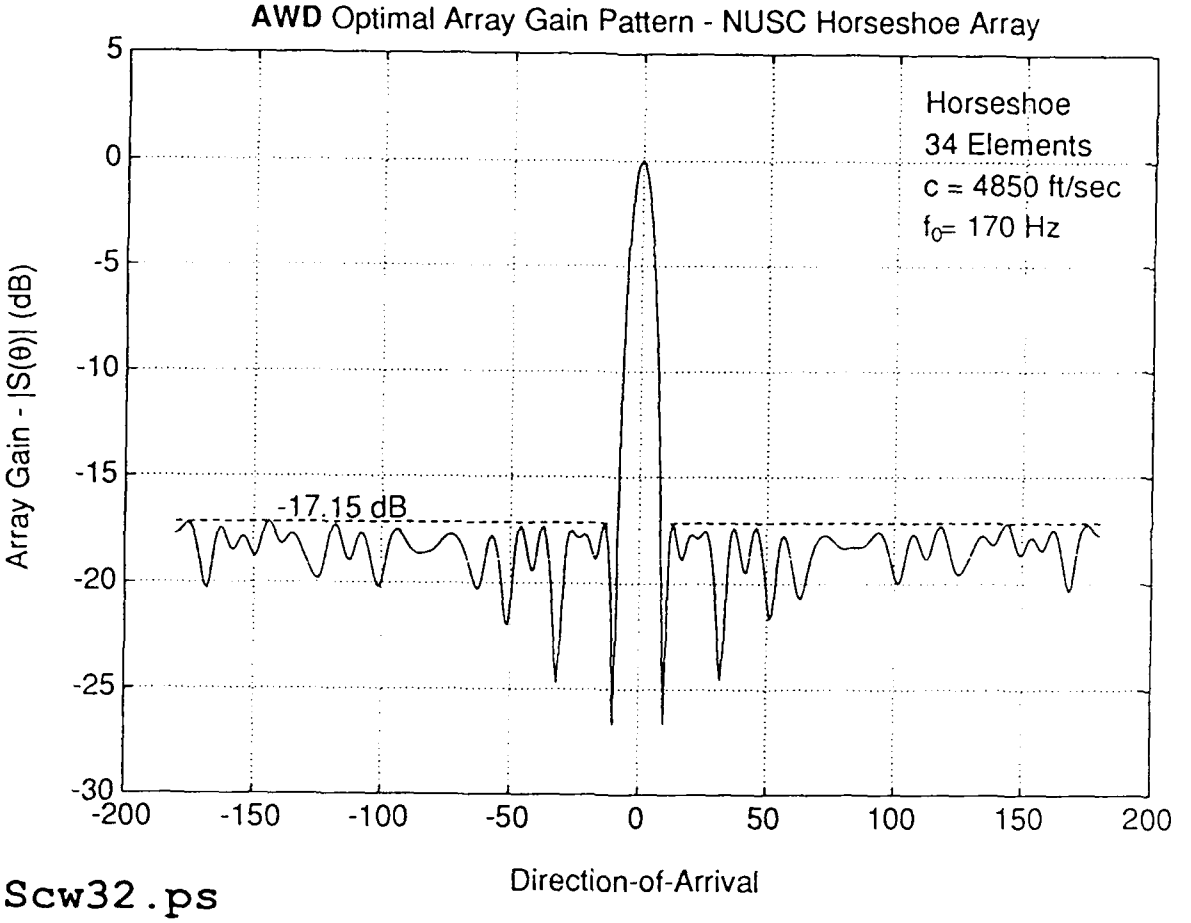
NUSC Optimal Array Gain - Design Frequency 170 Hz



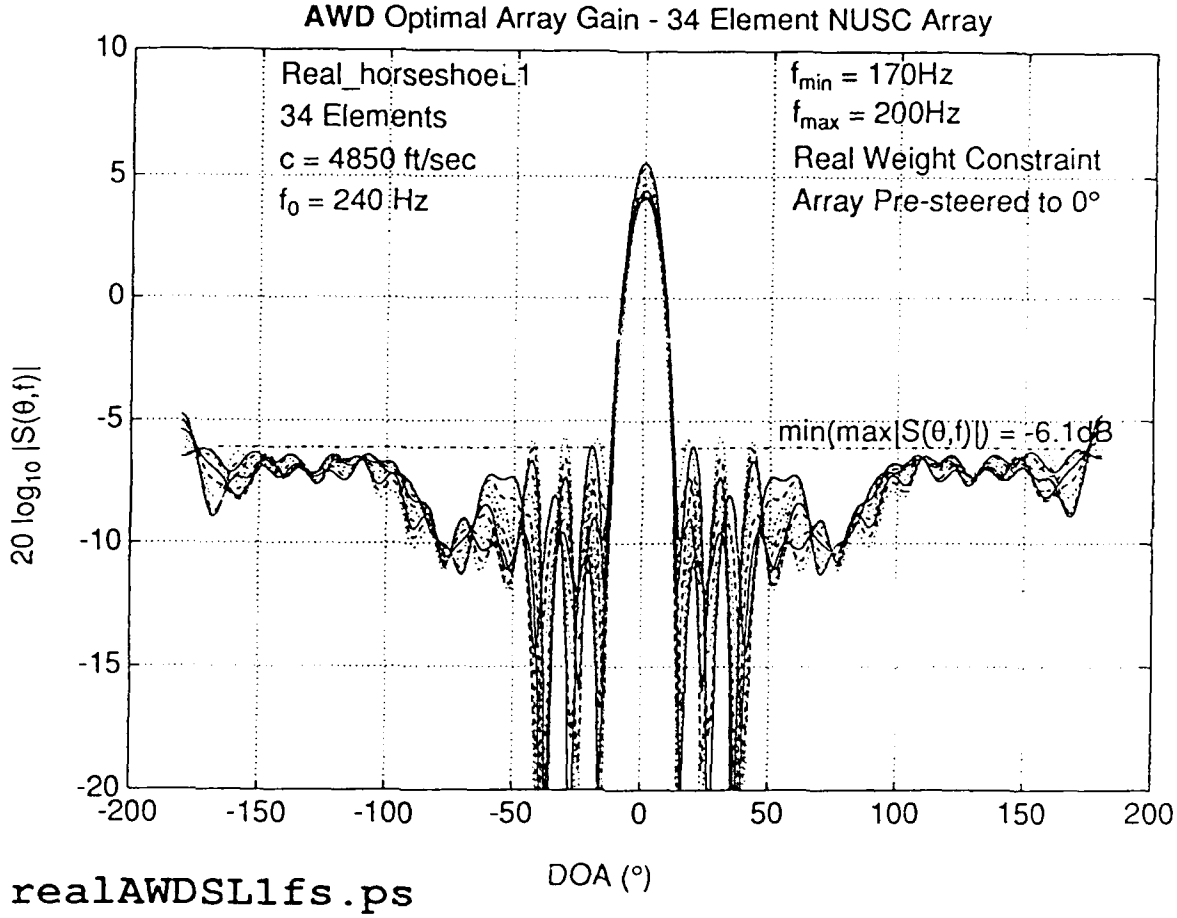
NUSCDOA0mesh1.ps



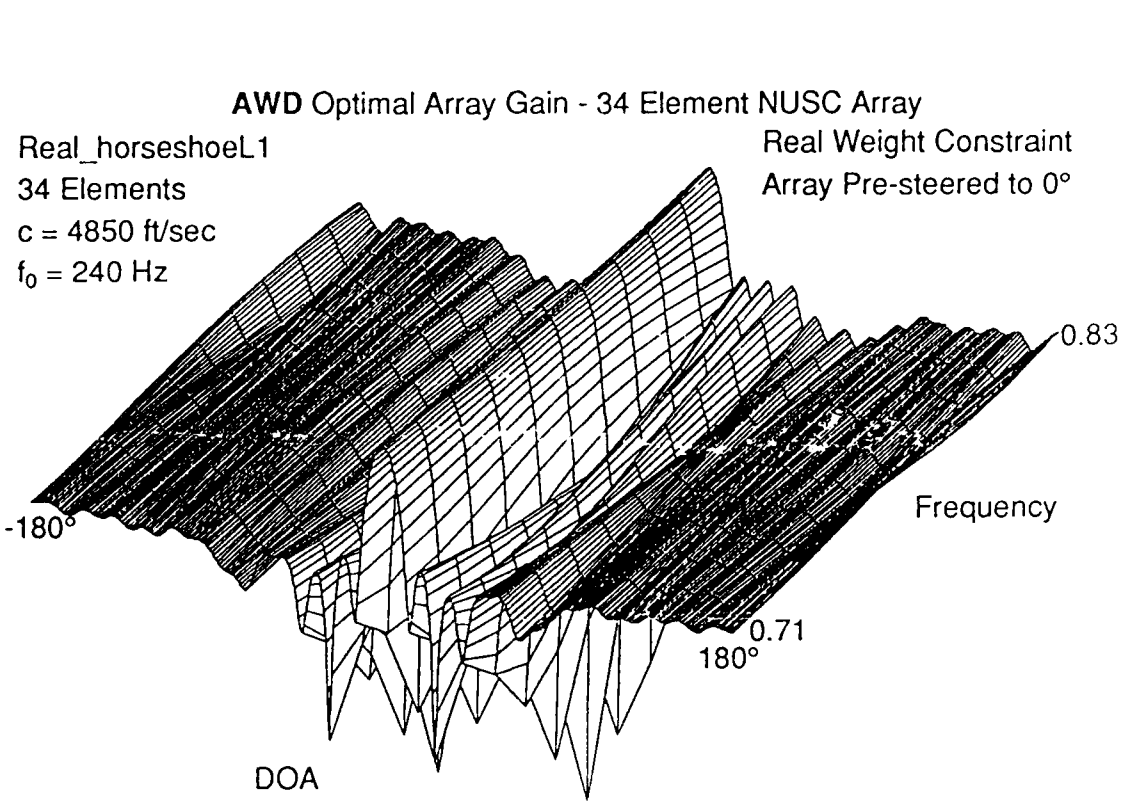
AWDNUSCdes3.ps


AWD Optimal Array Gain Pattern - NUSC Horseshoe Array
Horseshoe
34 Elements
c = 4850 ft/sec
f_0= 170 Hz
-17.15 dB
Array Gain - |S(θ)| (dB)
Direction-of-Arrival
5
0
-5
-10
-15
-20
-25
-30
-200
-150
-100
-50
0
50
100
150
200

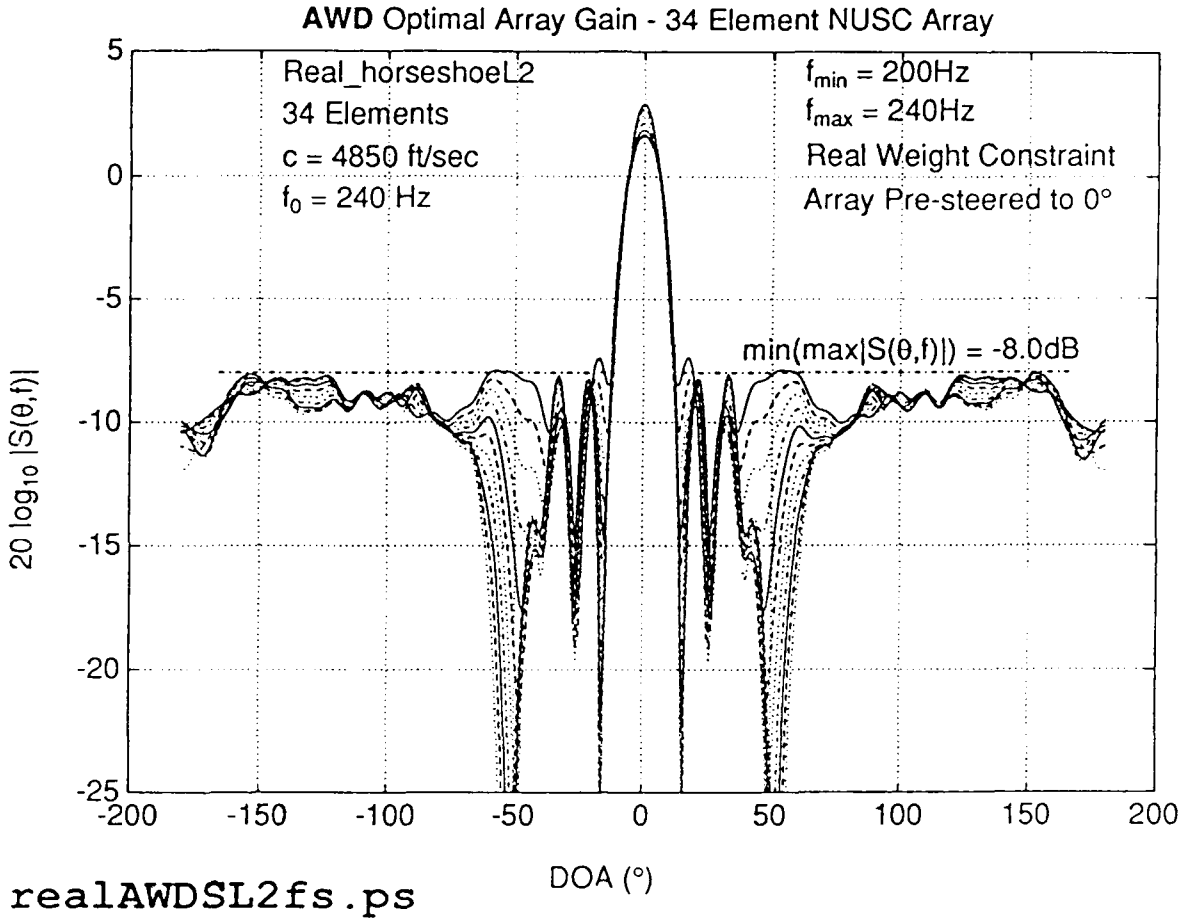
Scw32.ps



realAWDSL1fs.ps

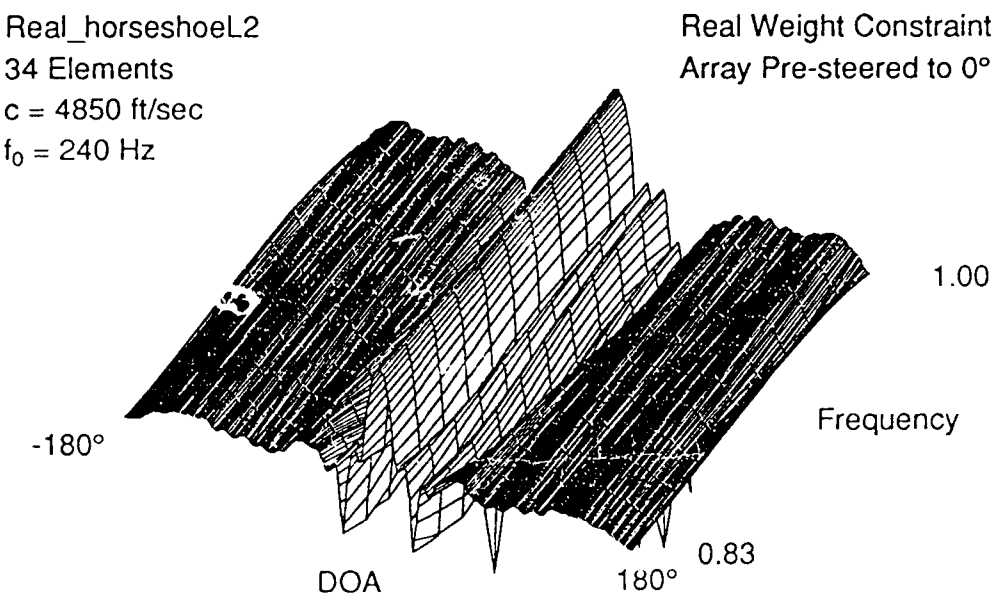


realAWDSL1mesh1.ps



realAWDSL2fs.ps

AWD Optimal Array Gain - 34 Element NUSC Array



realAWDSL2mesh1.ps

Prometheus Inc.
Final Report
11 July 1990

Appendix
Miscellaneous Information

PART I - THE SCHEDULE

SECTION B - SUPPLIES OR SERVICES AND PRICES/COSTS

0001 RESEARCH

The contractor shall furnish the level of effort specified in Section F, together with all related services, facilities, supplies and materials needed to conduct the research and prepare the reports described below. The research shall be conducted and reports delivered in accordance with Section F.

0001AA

1. Develop the Direct Adaptive Antenna System, incorporating analytic null steering methods drived during Phase I into existing adaptive algorithms.

2. Produce notch filters with more that one notch, similar to ideal single-notch filters developed in Phase I.

3. Construct and code wavelets (harmonics) for high frequency high resolution expansion of signal arising in time-varying environments.

4. Develop a convex programming approach to the problem of choosing antenna array weights.

5. Consider the development of an alternative to the Cox efficiency constraint for reducing the effect of shading coefficient errors.

6. Design and build a prototype compiler which will translate specifications stated in a natural language for antenna engineers (PSL) into executable code in the form of a convex program or an appropriate linear program.

7. Develop a Problem Scenario Generator to produce inputs to the executable code generated by the complier. This will include a Signal Generator and Operators capable of combining various signals.

8. Develop an Antenna Data Analyzer to generate plots illustrating the output of the executable code.

9. Develop a user Interface to enable the above components to be employed in an iterative and interactive manner.

10. Expand upon the new peak factor results obtained in Phase I by considering the general problem of determining, for a given amplitude spectrum, how to choose the phases of Fourier coefficients in order to minimize the peak factor in the corresponding inverse Fourier transform.

Prometheus Professional Personnel
Contract #F49620-88-C-0028

Abrahams, Paul W., Ph.D. in Mathematics, *MIT*, 1963.

Barrett, Michael J., Ph.D. in Electrical and Computer Engineering, *University of California, Santa Barbara*, 1980.

Benedetto, John J., Ph.D. in Mathematics, *University of Toronto*, 1964.

Boyd, Stephen P., Ph.D. in Electrical Engineering and Computer Science, *University of California, Berkeley*, 1985.

Byrnes, James S., Ph.D. in Mathematics, *Yeshiva University*, 1967.

Giroux, André, Ph.D. in Mathematics, *University of Montreal*, 1973.

Goldstein, Martin I., Ph.D. in Mathematics, *University of Wisconsin*, 1969.

Khraishi, Nasser, Ph.D. in Electrical Engineering, *Stanford University*, 1989.

Miller, Kenneth S., Ph. D. in Mathematics, *Columbia University*, 1950.

Newman, Donald J., Ph.D. in Mathematics, *Harvard University*, 1953.

Ostheimer, Gerald K., M.S. in Computer Science, *University of Massachusetts at Boston*, 1988.

Roy, Richard R., Ph.D. in Electrical Engineering, *Stanford University*, 1987.

Shapiro, Harold S., Ph.D. in Mathematics, *MIT*, 1952.

Recent Invited Lectures, James S. Byrnes

1. University of Montreal, October-December, 1989 (10 lectures)
2. University of New South Wales, February, 1990 (2 lectures)
3. George Mason University, March, 1990
4. University of Paris, Orsay, April, 1990
5. University of Pisa, May, 1990
6. University of Calabria, May, 1990
7. ETH-Zurich, May, 1990

Prometheus Publications
Contract #F49620-88-C-0028

1. J.S. Byrnes, The Minimax Optimization of an Antenna Array Employing Restricted Coefficients, *Scientia*, 1 (1988), 25-28.

2. J.S. Byrnes and D.J. Newman, Null Steering Employing Polynomials with Restricted Coefficients , *IEEE Transactions on Antennas and Propagation*, 36-2 (1988), 301-303.

3. J.S. Byrnes, A Notch Filter Employing Coefficients of Equal Magnitude, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36-11 (1988), 1783-1784.

4. J.S. Byrnes, Error Estimates Resulting from the Norms of Certain Noise Covariance Matrices, *J. of Sound and Vibration*, 136-3 (1990) 373-378.

5. D.J. Newman and J.S. Byrnes, The L^4 Norm of a Polynomial With Coefficients ± 1 (with D. J. Newman), *American Mathematical Monthly* 97-1 (1990) 42-45.

6. D.J. Newman and André Giroux, Properties on the Unit Circle of Polynomials with Unimodular Coefficients, *Proceedings of the American Mathematical Society* 109-1 (1990) 113-116.

7. J.S. Byrnes, Beamforming Applications of Polynomials with Restricted Coefficents, *Proc. of the 1987 NATO Advanced Study Institute on Electromagnetic Modeling and Measurements for Analysis and Synthesis Problems*, (1990), (to appear).

8. J.S. Byrnes, Problems on Polynomials with Restricted Coefficients Arising from Questions in Antenna Theory, *Proc. of the 1989 NATO Advanced Study Institute on Fourier Analysis and its Applications*, (1990) (to appear).

9. J.J. Benedetto, A Multidimensional Wiener-Wintner Theorem and Spectrum Estimation, *Transactions of the American Mathematical Society*, (1990), (to appear).

10. Nasser Khraishi, Richard Roy and Stephen Boyd, A Specification Language Approach to Solving Convex Antenna Weight Design Problems, (to be submitted).

11. J.J. Benedetto, Heisenberg Wavelets and the Uncertainty Principle, (to be submitted).

12. J.S. Byrnes, An Ideal Omnidirectional Transmitting Array, and Optimal Peak Factor Array, for Less Than Half-wavelength Spacing, (to be submitted).

REPORT OF INVENTIONS AND SUBCONTRACTS

(Pursuant to "Patent Rights" Contract Clause) (See Instructions on Reverse Side.)

Form Approved
OMB No. 0704-0240
Expire Sep 30, 1988

1a. NAME OF CONTRACTOR / SUBCONTRACTOR	c. CONTRACT NUMBER	2a. NAME OF GOVERNMENT PRIME CONTRACTOR	c. CONTRACT NUMBER	3. TYPE OF REPORT (X one)
Prometheus Inc.	F49620-88-C-0028	Same		a. INTERIM [ ] b. FINAL [X]
b. ADDRESS (Include ZIP Code)	**d. AWARD DATE (YYMMDD)**	**b. ADDRESS (Include ZIP Code)**	**d. AWARD DATE (YYMMDD)**	**4. REPORTING PERIOD (YYMMDD)**
103 Mansfield St. Sharon, MA 02067	871201			a. FROM 881201 b. TO 900331

SECTION I - SUBJECT INVENTIONS

5. "SUBJECT INVENTIONS" REQUIRED TO BE REPORTED BY CONTRACTOR / SUBCONTRACTOR (If "None," so state)

a. NAME(S) OF INVENTOR(S) (Last, First, MI)	b. TITLE OF INVENTION(S)	c. DISCLOSURE NO., PATENT APPLICATION SERIAL NO. OR PATENT NO.	d. ELECTION TO FILE PATENT APPLICATIONS				e. CONFIRMATORY INSTRUMENT OR ASSIGNMENT FORWARDED TO CONTRACTING OFFICER	
			(1) United States		(2) Foreign			
			(a) Yes	(b) No	(a) Yes	(b) No	(1) Yes	(2) No
None								

f. EMPLOYER OF INVENTOR(S) NOT EMPLOYED BY CONTRACTOR / SUBCONTRACTOR		g. ELECTED FOREIGN COUNTRIES IN WHICH A PATENT APPLICATION WILL BE FILED	
(1) (a) Name of Inventor (Last, First, MI)	(2) (a) Name of Inventor (Last, First, MI)	(1) Title of Invention	(2) Foreign Countries of Patent Application
(b) Name of Employer	(b) Name of Employer		
(c) Address of Employer (Include ZIP Code)	(c) Address of Employer (Include ZIP Code)		

SECTION II - SUBCONTRACTS (Containing a "Patent Rights" clause)

6. SUBCONTRACTS AWARDED BY CONTRACTOR / SUBCONTRACTOR (If "None," so state)

a. NAME OF SUBCONTRACTOR(S)	b. ADDRESS (Include ZIP Code)	c. SUBCONTRACT NO.(S)	d. DFAR "PATENT RIGHTS"		e. DESCRIPTION OF WORK TO BE PERFORMED UNDER SUBCONTRACT(S)	f. SUBCONTRACT DATES (YYMMDD)	
			(1) Clause Number	(2) Date (YYMM)		(1) Award	(2) Estimated Completion
None							

SECTION III - CERTIFICATION

7. CERTIFICATION OF REPORT BY CONTRACTOR / SUBCONTRACTOR (Not required if [X] Small Business or [ ] Non-Profit organization) (X appropriate box)

a. NAME OF AUTHORIZED CONTRACTOR / SUBCONTRACTOR OFFICIAL (Last, First, MI)	c. I certify that the reporting party has procedures for prompt identification and timely disclosure of "Subject Inventions," that such procedures have been followed and that all "Subject Inventions" have been reported.	
Byrnes, James S.		
b. TITLE	**d. SIGNATURE**	**e. DATE SIGNED**
President		

DD Form 882, NOV 87 *Previous editions are obsolete.* 882/317